UNIVERSIDAD AUTONOMA DE MADRID

Escuela Politécnica Superior

19|20

Trabajo fin de máster

Recomendador de contenidos de Movistar+

Beatriz Zurera Martínez-Acitores

Escuela Politécnica Superior
Universidad Autónoma de Madrid
C\Francisco Tomás y Valiente nº 11

excelencia Campus Internacional UAM CSIC+

www.uam.es

Universidad Autónoma de Madrid

Escuela Politécnica Superior

Proyecto para la obtención del título de
Máster en Ingeniería Informática
por la Universidad Autónoma de Madrid

Ponente del trabajo fin de máster:
Fernando Díez Rubio

Tutor del trabajo fin de máster:
Alejandro Bellogín Kouki
José Francisco Gazapo Aguirre

Recomendador de contenidos de Movistar+

Beatriz Zurera Martínez-Acitores

Beatriz Zurera Martínez-Acitores
Recomendador de contenidos de Movistar+

Beatriz Zurera Martínez-Acitores
C\ Francisco Tomás y Valiente Nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

Those who can imagine anything,
can create the impossible.

Alan Turing

Resumen

El consumo de vídeo, tanto de televisión como de contenido bajo demanda, se ha visto incrementado en los últimos años. Este acontecimiento está directamente relacionado con el incremento de las plataformas de contenido, como por ejemplo Netflix [1], HBO [2], Amazon Prime Video [3], Youtube [4], Apple TV [5], Disney TV [6], atresplayer [7], MiTele [8], etc. El surgimiento de estas plataformas ha provocado que la visualización de Video On-Demand (VOD) ya represente casi la mitad del consumo total de vídeo y televisión (TV).

Se ha visto que los usuarios no sólo están consumiendo más, sino que también están cambiando la forma de disfrutar los contenidos. En la actualidad, los usuarios tienen un abanico muy grande de posibilidades para acceder a los contenidos, a través de la televisión tradicional, del contenido bajo demanda, del contenido grabado o del disponible en modo sin conexión.

Ante una oferta tan amplia de contenidos es fácil que el usuario tenga problemas para encontrar lo que quiere visualizar. El descubrimiento de contenido es un desafío y los consumidores consideran que los métodos de descubrimiento actuales no son útiles. De hecho, según [9] los usuarios dedican una media de 30 minutos hasta encontrar el VOD que buscaban.

Motivado por lo expuesto anteriormente, en este trabajo se va a implementar una aplicación en HyperText Markup Language (HTML) para la recomendación de contenidos en la plataforma Movistar+. En esta aplicación el usuario podrá obtener recomendaciones personalizadas de contenidos en función de sus gustos y preferencias dentro de la plataforma.

Para llevar a cabo esta implementación se ha trabajo en tres líneas principales. Por un lado, se ha diseñado un Software Development Kit (SDK) dentro del dispositivo. Por otro, se ha realizado un estudio y una extracción de los datos que se tienen dentro de la plataforma. Y, por último, se ha desarrollado la interfaz desde la que el usuario podrá acceder a las recomendaciones. Esta implementación se ha basado en el algoritmo de recomendación de filtrado colaborativo basado en ítems, es decir, que realiza la recomendación sobre las puntuaciones de los contenidos dadas por otros usuarios de la plataforma.

Los resultados de las pruebas realizadas a cada una de las partes por separado y luego en su conjunto han sido analizados detalladamente para comprobar el comportamiento de la aplicación en un entorno con usuario reales. Las respuestas obtenidas de las pruebas realizadas dentro del entorno de preproducción han sido satisfactorias y demuestran que se podría llevar a cabo un proyecto de este tipo en la plataforma para aumentar la satisfacción de los clientes.

Palabras clave

Sistema de recomendación, algoritmo, filtrado colaborativo, algoritmo basado en ítem, consumo de vídeo, SDK, API, IPTV, VOD, REST.

Abstract

Video consumption, both linear television and On-Demand content, has increased in recent years. This boost is directly related to the increment in the number of video streaming services, such us Netflix [1], HBO [2], Amazon Prime Video [3], Youtube [4], Apple TV [5], Disney TV [6], atresplayer [7], MiTele [8], etc. The emergence of these streaming platforms has caused that the display of VOD already represents almost half of the total of video and television consumption.

It has been noted that users are not only consuming more, but also changing the way they enjoy the content. Currently, users have a very wide range of possibilities to access video content, such as traditional television, VOD, recorded content, or offline content.

It is easy for the user to have trouble finding what they want to see given such a wide range of video content. Content discovery is a challenge and consumers believe that current discovery methods are not useful. In fact, according to [9] users, they spend an average of 30 minutes until they find the VOD they are looking for.

Motivated by this, in this Thesis an HTML application that recommends VOD in the Movistar+ servide will be implemented. In this application, the user will obtain personalised recommendations based on their preferences within the service.

The implementation has been carried out on three main lines. Firstly, an SDK for the hardware device has been designed. Additionally, a study of the data within the service and how to extract it has been performed. And, finally, the web interface from which the user will have access to the recommendations has been developed. This implementation has been based on the Collaborative Filtering (CF) algorithm based on items, that is, it provides recommendations according to the item score given by other users within the service.

The results of the tests performed, individually on each part and as a whole, have been analysed in detail to determine the behaviour of the application in a real environment. The responses obtained from the tests conducted in the pre-production environment have been satisfactory and demonstrate that such project could be carried out on the service, with a result of increased customer satisfaction.

Keywords

Recommender system, algorithm, collaborative filtering, item-based algorithm, video consumption, SDK, API, IPTV, VOD, REST.

Índice

Listas

Lista de códigos

Lista de ecuaciones

Lista de figuras

Lista de tablas

1

Introducción

En este proyecto se plantea realizar una aplicación para la recomendación de contenidos de la plataforma de Movistar+, para lo cual, como se detalla más adelante, se han analizado los algoritmos de recomendación, se han estudiado los datos disponibles en la plataforma y se ha implementado la aplicación. Los datos creados y obtenidos durante la ejecución de este proyecto han sido extraídos de líneas de usuarios que han dado su consentimiento para este prueba de concepto.

En este capítulo se va a explicar en primer lugar la motivación por la cual se ha decidido realizar este proyecto. Después se van a enunciar los objetivos que se han pretendido alcanzar para el proyecto. Por último, se va a exponer la estructura de esta memoria, de forma que el lector pueda comprender rápidamente qué se va a explicar en cada una de las secciones.

1.1. Motivación

En los últimos años el consumo de vídeo y las plataformas de vídeo han sufrido una gran evolución que ha permitido que se pueda disfrutar del acceso al vídeo desde casi cualquier parte. El aumento del consumo de vídeo se refleja en los informes sobre la televisión, contenido digital y medios de comunicación que se vienen haciendo por empresas independientes en los últimos años.

Por ejemplo, según el marco general de medios de comunicación realizado en 2017 [10] el sitio web más visitado es la página Youtube [4] con una penetración de más del 60 %. Por otro lado, en el informe de 2017 sobre televisión y medios de comunicación realizado por Ericsson [9], se observan datos como los siguientes:

- El 70 % de los usuarios encuestados en 2017 visualiza los vídeos en un smartphone, lo que supone el doble que en 2012. De hecho, en este informe [11] se constata que en 2018 el 60 % del tráfico de datos móviles provienen de la visualización de vídeos.
- Para el 2020 se espera que la media de visualización de VOD supere las 25 horas semanales entre los jóvenes de entre 16 y 19 años. Esto implica un incremento del 180 % desde 2010.

- Para el 2020 se estima que la mitad de las visualizaciones se realizarán en una pantalla móvil, lo que significa un aumento del 58 % desde 2010.

La recomendación de contenidos sigue siendo un reto y, según los usuarios encuestados [9], los sistemas de recomendación no son todo lo útiles que los usuarios esperan. La media total para la búsqueda de un contenido se ha visto incrementada en los últimos años:

- En 2016 un usuario dedicaba una media de 45 minutos a localizar un contenido para visualizar, distribuídos de la siguiente manera 24 minutos para contenidos de TV y 21 minutos para VOD.
- En 2017 un usuario dedica una media de 51 minutos a realizar la misma tarea, 24 minutos para contenidos de TV y 27 minutos para VOD.

En base a estos últimos datos se puede percibir que la gente se siente perdida entre tanto contenido, ya que, como también recoge el informe de Ericsson [9], 6 de cada 10 usuarios creen que el descubrimiento de contenidos es la clave a la hora de suscribirse a un nuevo servicio de TV y de VOD. Este trabajo pretende enfocarse en el desarrollo de una aplicación de recomendación de contenidos para aumentar la satisfacción del cliente.

El departamento con el que se ha trabajado para la realización del Trabajo Fin de Máster (TFM) se centra en el desarrollo de los servicios Internet Protocol Television (IPTV) de Movistar+. Desde Movistar se hace llegar a los clientes un Set-Top Box (STB) para disfrutar de los contenidos y los servicios que ofrece la plataforma. El STB es un receptor de señales digitales que convierte las señales analógicas en señales digitales y permite reproducirlas en la TV. Además, a lo largo de los años se ha ido mejorando la interfaz gráfica adecuándose a las peticiones de los usuarios e incorporando nuevas funcionalidades que mejoran la experiencia de este.

1.2. Objetivos

En este apartado se definen los objetivos concretos que se pretenden alcanzar en este proyecto y el procedimiento que se ha seguido para conseguirlo.

Por claridad, se ha dividido este proyecto en tres objetivos generales, los cuales luego se dividirán a su vez en objetivos específicos. Los objetivos generales son:

OG-1 Diseño e implementación de un Application Programming Interface (API) que permita a proveedores externos acceder a la interfaz del dispositivo.

OG-2 Estudio y análisis de los algoritmos de recomendación y de los datos que se disponen dentro de la plataforma, para distinguir aquellos que realmente sean útiles para el algoritmo.

OG-3 Desarrollo de la interfaz de la aplicación donde se van a mostrar los contenidos reco-

mendados en el STB que el usuario tenga en su domicilio.

El primer objetivo - OG-1 - consiste en diseñar un método de comunicación entre la aplicación principal y otras aplicaciones. Esta comunicación dará la posibilidad de crear líneas de desarrollo paralelas, sin necesidad de interferir en los desarrollos de la aplicación principal. Estas aplicaciones secundarias corren en un navegador HTML por lo que las aplicaciones cuentan con la restricción de tener que implementarse en HTML. Este navegador viene instalado en el STB y es capaz de invocar a ciertas capacidades del mismo, como los gráficos, el vídeo, el audio, los datos del usuario, el mando a distancia, etc.

Para realizar el diseño y la implementación, primero se realizará un estudio de las llamadas que se realizan desde el STB al Back-End (BE). Después, se investigará en las capacidades que presenta el navegador instalado en el dispositivo. Y, por último, se realizarán pruebas unitarias y de integración para comprobar el desarrollo implementado.

OG-1 Diseño e implementación de un API que permita a proveedores externos acceder a la interfaz del dispositivo.

- **OE-1.1** Análisis de las llamadas realizadas al BE.
- **OE-1.2** Estudio del navegador utilizado por el proveedor.
- **OE-1.3** Desarrollo, integración y pruebas.

En el segundo objetivo - OG-2 - se pretende realizar un estudio de los algoritmos de recomendación que existen actualmente. Además, se realizará una selección del algoritmo más apropiado en función de los datos disponibles que están en la plataforma. Para ello, se estudiarán los diferentes datos disponibles en la plataforma para realizar una extracción de los mismos. Se recuerda que los datos con los que se ha trabajado durante este proyecto han sido creados y extraídos de líneas de usuarios que han dado su consentimiento para este fin. Se definen para ello los siguientes objetivos específicos:

OG-2 Estudio y análisis de los algoritmos de recomendación y de los datos que se disponen dentro de la plataforma, para distinguir aquellos que realmente sean útiles para el algoritmo.

- **OE-2.1** Investigación sobre algoritmos de recomendación.
- **OE-2.2** Estudio de los datos disponibles en la plataforma y extracción de los datos útiles para los algoritmos.
- **OE-2.3** Selección del algoritmo de recomendación, de acuerdo a los datos extraídos.

Por último, en el último objetivo general - OG-3 - se trata de realizar el desarrollo de la aplicación para la recomendación de contenidos dentro de la plataforma de Movistar+. Para el desarrollo de

esta aplicación web hay que realizar un estudio de las limitaciones hardware que tiene el dispositivo, como la velocidad de carga, el rendimiento, el almacenamiento, las animaciones, etc. Finalmente, para comprobar que la aplicación funciona correctamente en el STB, se realizarán unas pruebas en diferentes dispositivos.

OG-3 Desarrollo de la aplicación donde se van a mostrar los contenidos recomendados desde el STB que el usuario tenga en su domicilio.

OE-3.1 Estudio de las limitaciones del hardware del STB.

OE-3.2 Desarrollo de la aplicación.

OE-3.3 Integración y pruebas.

1.3. Organización de la memoria

Una vez explicada la motivación y los objetivos de este proyecto, se va a explicar cómo se distribuye el resto de la memoria, de forma que quede claro qué se puede encontrar de manera general en cada una de las secciones.

En el Capítulo 2 de la memoria se habla del estado del dispositivo hardware, de los diferentes algoritmos y de las tecnologías que tienen influencia en este proyecto.

En el Capítulo 3 se definen en detalle el diseño de los datos de la plataforma, de los algoritmos de recomendación y de la interfaz de la aplicación desde la que se mostrará el resultado.

En el Capítulo 4 se explican los detalles del procedimiento seguido para la extracción de los datos de la plataforma, del desarrollo del algoritmo y de la interfaz de la aplicación. Para ello, se da una idea general del código utilizado y de los escenarios que han servido para comprobar su funcionamiento.

En el Capítulo 5 se presentan y analizan los resultados de cada una de las partes diseñadas. El objetivo es comprobar el funcionamiento de un escenario real punto a punto con la integración de todas las piezas que se han implementado en fases anteriores.

En el Capítulo 6 se realiza una conclusión general del trabajo. En ella se pretende resumir lo más reseñable que se ha podido observar a lo largo del proyecto. Además, también se puede encontrar el trabajo futuro del proyecto.

Por último, se incluyen la bibliografía, la lista de definiciones, la lista de acrónimos y los anexos. Dentro de los anexos se puede encontrar la explicación detallada de la planificación que se ha seguido dentro del proyecto.

2

Estado del arte

En este capítulo se pretende exponer el estado y el marco teórico de los algoritmos y las tecnologías más importantes que incluye este proyecto. De esta forma, se explican parte de los fundamentos sobre los que se sostiene este TFM.

En el apartado 2.1 se explica la metodología utilizada en el proyecto. Después, en el apartado 2.2 se detallan brevemente diferentes tecnologías y herramientas en las que ha basado el proyecto. Luego, en el apartado 2.3 se muestra la evolución de los dispositivos hardware usados en IPTV. Y, por último, el apartado 2.4 explica los sistemas de recomendación y hace un repaso de los que están disponibles en estos momentos.

A continuación, se realiza una breve explicación de cada uno de ellos. Se pretende responder a preguntas como qué son, por qué surgen, en qué se fundamentan y cuáles son sus ventajas e inconvenientes.

2.1. Metodologías ágiles

En este trabajo se ha seguido una metodología ágil muy conocida llamada SCRUM [12]. SCRUM es un marco de trabajo iterativo e incremental para el desarrollo de proyectos, productos y aplicaciones [13].

En SCRUM el proceso ágil se divide en diferentes ciclos de trabajo, llamados sprints, que se componen de planificación, desarrollo, test y revisión. Los productos obtenidos al finalizar el sprint son productos funcionales, lo que significa que el código está integrado, completamente probado y listo para entregar. En la Figura 2.1 se muestra de manera gráfica el proceso completo.

Estos ciclos de trabajo se repiten hasta que el producto está completo. Cada iteración tiene una duración de entre 1 y 4 semanas, y se desarrolla una detrás de otra. Los sprints no se alargan de la fecha establecida, sino que terminan en una fecha específica aunque no se haya terminado el trabajo.

Al comienzo de cada sprint el equipo planifica lo que se va a llevar a cabo durante ese sprint; es decir, durante el sprint no se puede cambiar esta planificación. Todos los días el equipo se reúne breve-

mente para informar del progreso. Al final del sprint, el equipo revisa dicho sprint con los interesados en el proyecto, y les enseña lo que han construido. Los interesados realizan comentarios y observaciones que se pueden incorporar al siguiente sprint.

Los roles principales en Scrum son:

- **Scrum Master** que mantiene los procesos y trabaja junto con el jefe de proyecto.
- **Product Owner** que representa a las personas implicadas en el negocio.
- **Team** que incluye a los desarrolladores.

Una parte muy importante de SCRUM son las reuniones que se realizan durante cada una de las iteraciones. Hay distintos tipos:

- **Scrum diario:** cada día durante la iteración tiene lugar una reunión de estado del proyecto.
- **Reunión de planificación de iteración (sprint):** se lleva a cabo al principio del ciclo de la iteración.
- **Reunión de revisión de iteración:** al final del ciclo de la iteración.
- **Iteración retrospectiva:** al final del ciclo de la iteración.

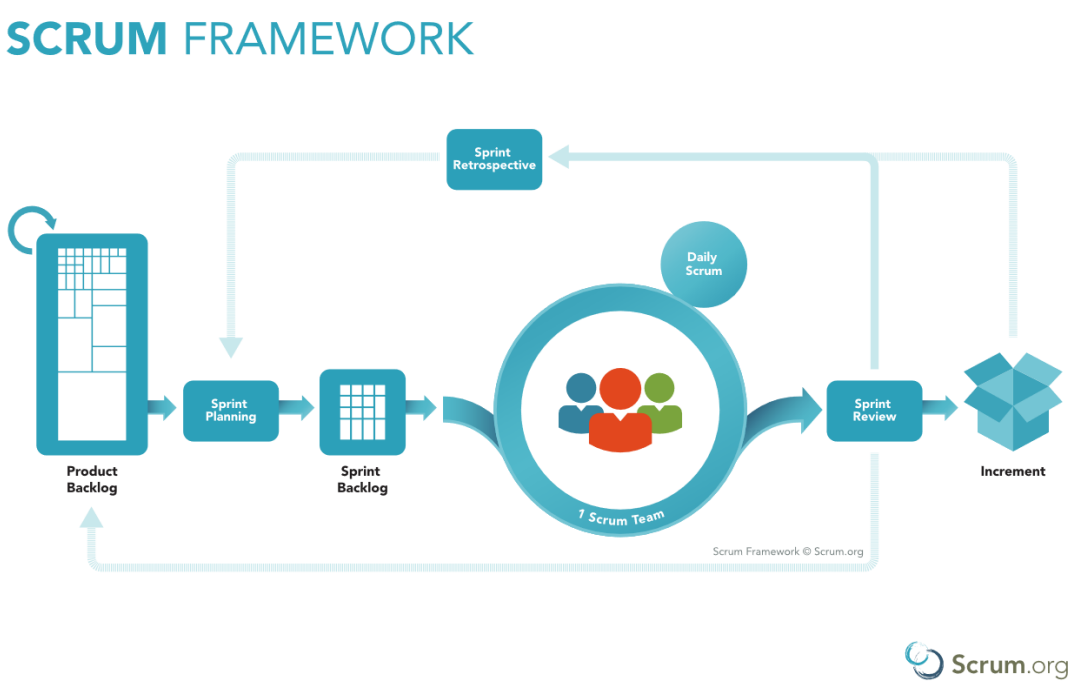

Figura 2.1: Figura que muestra el proceso de SCRUM [13].

En este proyecto hemos seguido este marco de trabajo a través de las reuniones mencionadas anteriormente. Para las reuniones de planificación de cada sprint, además de las presenciales, se han utilizando diferentes tecnologías: algunas se han realizado en forma de videoconferencias gracias a la

herramienta de Hangouts [14] y otras en forma de correo electrónico.

2.2. Tecnologías

2.2.1. IPTV

En este apartado se define y se pretende dar una idea del concepto de IPTV. Para entender de una manera más clara este concepto, antes se procede a explicar brevemente las redes de distribución. Por un lado, tenemos las redes de difusión y, por otro, las redes de internet. Éstas a su vez se dividen en las siguientes:

- **Redes de difusión:** emiten el contenido desde un punto central permitiéndole sintonizar canales específicos dentro de la señal.
  - TV por cable: se transmite a través de una red de cable coaxial dedicada.
  - TV por satélite: se emplean ondas de radio que se reciben mediante antena parabólica de manera inalámbrica.
  - Television Digital Terrestre (TDT): la señal se transmite a través de una red de repetidores terrestres.
- **Redes de internet:** funcionan gracias al Internet Protocol (IP), transfiriendo paquetes desde el servidor al dispositivo del usuario final. Esto ha supuesto un cambio radical en la forma en que las personas estaban acostumbradas a consumir contenidos de vídeo.
  - Over-The-Top (OTT): consiste en la transmisión de vídeo junto a otros servicios.
  - IPTV: se distribuye la señal de TV por protocolo IP a través de conexiones de banda ancha.

Accesibilidad de las redes de distribución

Este proyecto se ha basado en las redes de internet, por lo que se profundiza un poco más sobre estas dos tecnologías a continuación:

OTT

La tecnología OTT está basada en una conexión a internet de acceso público compartiendo la red con el resto del tráfico. Al tener un coste de infraestructura reducido, los precios de servicio son también más accesibles. Además, tienen otra gran ventaja que es la sencillez de instalación, en tanto que lo único que se necesita es un dispositivo conectado a internet.

Por contra, la calidad del servicio está sujeta a la congestión de la red en ese momento. Adicional-

mente, esta congestión puede impactar en la latencia de servicio por no tener un tráfico priorizado.

IPTV

En esta tecnología se usa una red de distribución privada y administrada por un operador, de esta manera al tener un ancho de banda dedicado únicamente para el vídeo se puede garantizar la calidad tanto de la imagen como del sonido.

La desventaja de esta tecnología es que requiere un dispositivo que decodifique la señal, que se verá más adelante.

Como se ha visto en este apartado, la diferencia entre las redes de difusión y las redes de internet es su medio de distribución. Además, dentro de las redes de internet la gran diferencia entre IPTV y OTT es que en la primera se sirve la señal con un ancho de banda reservado, garantizando la calidad de la señal y la distribución. Por el contrario, la segunda no utiliza una red gestionada para el vídeo, sino que se comparte la red con el resto de tráfico.

2.2.2. API

API [15] son las siglas de Application Programming Interface. Es un interfaz que permite la comunicación entre dos sistemas sin necesidad de conocer la implementación de cada uno de ellos. Este mecanismo recoge peticiones del sistema emisor, llamado sistema consumidor, las transforma y se las envía al sistema receptor. Una vez el sistema receptor, o sistema proveedor, tiene una respuesta a esa petición el mecanismo recoge la respuesta y la envía de vuelta al receptor. Gracias a esta interfaz se puede acceder a la información de uno de esos sistemas de manera sencilla. Se puede definir también como un contrato entre una aplicación de software y un servicio. Normalmente la aplicación de software toma el rol de consumidor del API, mientras que el servicio es el proveedor.

Existen tres enfoques respecto a las políticas de las versiones de las API [15]:

- **Privado:** estas API se desarrollan dentro de una empresa y se explotan de manera interna. Esto ofrece a las empresas un mayor control sobre ellas.
- **De socios:** las API se comparten con socios específicos. Esto puede proporcionar flujos de ingresos adicionales, sin comprometer la calidad.
- **Público:** en este caso, las API se encuentran accesibles públicamente, por ejemplo en Internet. Esto permite a terceros, con los que no se tiene ningún contrato o ninguna interacción, puedan consumir nuestras API, o nos permite acceder a las API y datos de terceros sin necesidad de un contrato o una relación con el desarrollador.

En este proyecto se ha desarrollado un API privado, por eso se mantendrá la confidencialidad empresarial sobre este apartado.

2.2.3. Lenguajes de programación

En este proyecto se han utilizado, principalmente, tres lenguajes de programación que se detallan a continuación:

JAVA

Java [16] [17] es un lenguaje de programación orientado a objetos, fue desarrollado por la empresa Sun Microsystem a principios de los años 90. Sun Microsystem en 2007 liberó la mayor parte de sus tecnologías Java bajo la licencia GNU GPL. Java toma mucha de su sintaxis de C y C++, por lo que un programador que haya aprendido cualquiera de estos dos lenguajes encontrará relativamente sencillo aprender a programar en Java. Aunque tome mucha de la sintaxis de C y C++ elimina alguna de sus características de más bajo nivel, como son los punteros y la manipulación directa de la memoria. Con respecto a la memoria, su gestión no es un problema ya que ésta es gestionada por el propio lenguaje y no por el programador.

Se ha utilizado el lenguaje Java para la programación del sistema de recomendación al utilizar la librería RankSys. También se ha utilizado para crear el servidor que sirve las recomendaciones de los usuarios. Este servidor se ha montado usando Dropwizard [18], que es un framework Java para desarrollar rápidamente servicios Representational State Transfer (REST) de alto rendimiento.

HyperText Markup Language, Cascading Style Sheets y JavaScript

Por un lado, HTML [19] se conoce por sus siglas en inglés HyperText Markup Language. Es un lenguaje de marcado que utiliza etiquetas para identificar los diferentes tipos de contenido, a diferencia de los lenguajes de programación que utilizan funciones. Además, provee una estructura básica de las páginas web, que se modifica y se mejora con otras tecnologías como las siguientes.

Por otro lado, Cascading Style Sheets (CSS) es un lenguaje de estilo que indica el diseño y la presentación que tienen que tomar los elementos HTML. Se puede definir como el responsable de la presentación, el formato y el diseño de la página web. Por defecto, CSS no tiene lógica como los lenguajes de programación, simplemente presentación. Aunque hay alguna tecnología como Sass/-Less [20] [21] que permiten añadir funcionalidad a CSS.

Y, por último, JavaScript (JS) [22] se define como un lenguaje de programación interpretado de alto nivel que maneja el comportamiento de los elementos que se encuentran en la página web.

En resumen: por un lado, HTML provee las herramientas necesarias para estructurar el contenido en un sitio web. Por otro lado, CSS ayuda a dar forma a este contenido para que le muestre al usuario tal y como fue diseñado. Para terminar, JS proporciona la capacidad de dotar de interactividad a la página web.

2.3. Dispositivos hardware

El dispositivo hardware que se utiliza en IPTV es un STB, que, como ya se ha comentado, es un pequeño ordenador que se encarga de convertir la señal analógica en una señal digital.

Esto dispositivos han ido evolucionando [23] [24] según el desarrollo de las redes de distribución. En un principio estos dispositivos convertían la señal de vídeo analógica que se recibía por cable a una señal digital capaz de mostrarse por pantalla. Después convertían las señales de vídeo recibidas por satélite, luego por repetidores terrestres, hasta que finalmente son transmitidas por IP.

Al igual que los dispositivos, los servicios que éstos ofrecen también han evolucionado, ofreciendo cada vez más servicios adicionales como interactividad, alquiler/compra de contenido, grabación de los canales de TV, etc.

La gama de dispositivos IPTV se ha ido actualizando a lo largo de los años para soportar las nuevas tecnologías. En las Figuras 2.2(a) y 2.2(b) se encuentran los dispositivos que están descatalogados y que están siendo sustituidos poco a poco. Eran unos dispositivos muy básicos que no ofrecían casi servicios. Después, en la Figura 2.2(c) se pueden ver los dispositivos que transmiten el vídeo con una calidad SD y HD. Por último, en la Figura 2.2(d) se pueden ver los dispositivos que transmiten el vídeo con una calidad hasta UHD.

(a) Dispositivos descatalogados, como por ejemplo ADB3800.

(b) Dispositivos legados, como por ejemplo ADB2840 y Zyxel.

(c) Dispositivos SD y HD, como por ejemplo Vestel y Arris.

(d) Dispositivos UHD, como por ejemplo Proteus y Humax.

Figura 2.2: En estas imágenes se puede ver la evolución de los dispositivos en Movistar+.

Al igual que los equipos tecnológicos, los dispositivos IPTV han ido evolucionando sus capacidades. En la Figura 2.3 se muestran las mejoras en cuanto a capacidades en los últimos dispositivos instalados en los hogares de los clientes. Esta evolución de las capacidades conlleva, entre otras cosas, mejoras sustanciales en el rendimiento y en la velocidad de los dispositivos. A pesar de las diferencias, se intenta que la experiencia de usuario y los servicios ofrecidos sean los mismos en todos los dispositivos. Aunque, por ejemplo, los más nuevos tendrán mayor fluidez en la navegación por la interfaz que los más antiguos.

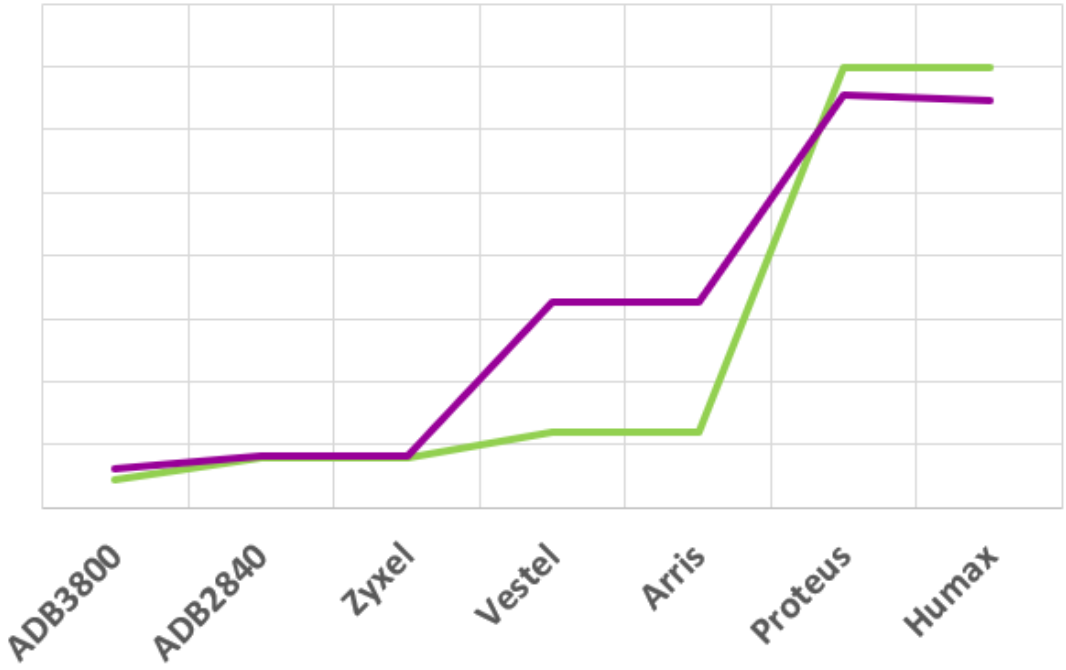

Figura 2.3: En esta figura se muestra la evolución de las capacidades de los Set-Top Box.

2.4. Sistemas de recomendación

El objetivo principal de los sistemas de recomendación es guiar al usuario a encontrar artículos (ítems) dentro de un gran catálogo, teniendo en cuenta sus preferencias. En los últimos años, los sistemas de recomendación han tomado gran valor debido al crecimiento exponencial de los catálogos de contenidos. Los sistemas de recomendación [25] [26] más utilizados se pueden dividir en varios tipos:

- **Basado en contenido (Content-based filtering (CBF))** sugiere al usuario ítems similares a aquellos que le gustaron en el pasado.
- **Filtrado colaborativo (CF)** sugiere al usuario ítems que han gustado a gente con gustos similares al suyo.
- **Filtrado híbrido (Hybrid Filtering (HF))** surge para intentar unir todas las ventajas de los anteriores y reducir así sus desventajas.

El dilema de los sistemas de recomendación es acertar con las predicciones de los ítems que se le muestran al usuario. Esas predicciones se obtienen en base a la información histórica almacenada por el sistema sobre el usuario. La información que se almacena del usuario pueden ser valores numéricos conocidos como ratings (valores ordinales que representan los niveles de interés del usuario sobre los ítems) o valores binarios.

Por otro lado, esta información se puede obtener del usuario de manera explícita (a través de, por ejemplo, ratings o reseñas que el usuario deja en el sistema) o de manera implícita (gracias a su histórico de visualizaciones o de compras).

En los siguientes apartados se describen los enfoques de cada uno de los principales sistemas de recomendación.

2.4.1. CBF

Estos sistemas de recomendación se basan en ítems similares a los que le han gustado al usuario en el pasado. Esta similitud se calcula haciendo coincidir los atributos del perfil del usuario, basado en sus gustos y preferencias, con los atributos de los ítems. Por ejemplo, si un usuario ha puntuado muy bien películas de terror el sistema de recomendación le devolverá contenidos de ese género.

Las limitaciones [26] [27] que presenta este modelo son las siguientes:

- **Análisis de contenido restringido**: CBF depende de los atributos asociados que tengan los ítems.
- **Nuevo usuario**: CBF puede recomendar a un usuario si éste tiene puntuaciones para suficientes ítems. Por eso a un nuevo usuario, que no tiene ninguna recomendación, no se le puede realizar una predicción.
- **Especialización extrema**: CBF recomienda ítems similares a los que el usuario ya ha puntuado, por lo que sufre el problema de exceso de especialización.
- **Falta de novedad**: asociado a la limitación anterior, los ítems recomendados son muy similares entre sí, lo que produce un sistema insuficiente en cuanto a diversidad o sugerencias redundantes.

2.4.2. CF

El filtrado colaborativo [28] [29] utiliza la puntuación de otros usuarios e ítems en el sistema. La idea principal es que, teniendo dos usuarios similares, es decir, que han puntuado de manera similar en el pasado, el sistema realizará recomendaciones al segundo usuario basándose en las puntuaciones que haya dado el primer usuario a aquellos ítems desconocidos para el segundo. Para determinar si dos usuarios son similares o no, estos sistemas tienen en cuenta ítems (vistos, comprados, consumidos, ...) de los usuarios y cómo los han valorado.

Este filtrado aborda las limitaciones comentadas en el modelo anterior. Por un lado, si un ítem no tiene atributos asociados pero ha sido puntuado por otro usuario puede aparecer en el listado de recomendación de un usuario similar. Además, el filtrado colaborativo mantiene el factor sorpresa en

las recomendaciones, ya que puede recomendar ítems muy variados, siempre y cuando haya llamado la atención de otros usuarios.

CF también tiene limitaciones [26] [27]:

- **Nuevo usuario**: al igual que CBF, este método también necesita que un usuario puntúe suficientes ítems para que se le pueda recomendar.
- **Nuevo ítem**: CF recomienda aquellos que han sido valorados, por lo que un nuevo ítem no puede ser recomendado porque no tiene puntuaciones.
- **Escasez de datos**: ocasionado por el pequeño porcentaje de puntuaciones del usuario frente a la cantidad de ítems disponibles.

Dentro del filtrado colaborativo encontramos las estrategias **basadas en usuarios** y las **basadas en ítems**. En la primera el sistema recomienda al usuario los ítems que han gustado a usuarios similares. Por el contrario, en el segundo, el sistema recomienda al usuario los ítems que guardan relación a los ítems que le han gustado previamente.

2.4.3. HF

Para superar ciertas limitaciones de los métodos anteriores, surge el enfoque de los sistemas de recomendación híbridos [30] [26] que combinan las características de ambos. El HF combina los modelos de CBF y CF usando las ventajas de CBF para solventar las limitaciones de CF. Por ejemplo, como se ha comentado CF presenta problemas con ítems nuevos o con ítems que no tienen puntuaciones, pero esto no limita el CBF ya que sus predicciones se basan en los atributos de los ítems. Estos dos modelos se pueden combinar de diferentes maneras, por ejemplo, pueden combinar sus predicciones en un única recomendación más robusta o se puede añadir más información al modelo CF.

Por otro lado, muchos enfoques existentes se centran en recomendar los ítems más relevantes para el usuario y no tienen en cuenta el contexto del usuario que está haciendo la petición en ese momento. Por ejemplo, en la recomendación de contenidos sería recomendable dotar al usuario de un contexto temporal [31] [32]. Las recomendaciones que se devuelvan al usuario no deberían ser iguales por la mañana que por la noche, ni en invierno o en verano, etc. De hecho, se ha demostrado en [33] que el enfoque del HF provee recomendaciones más certeras que los otros modelos por separado. Este enfoque es el que se encuentra entre los recomendadores más conocidos junto con inteligencia artificial y aprendizaje automático, entre ellos se encuentra Netflix [34] [35] [36] [37] [38], Spotify [39], Amazon [40] [41], etc.

2.4.4. Selección del algoritmo de recomendación

La selección del sistema de recomendación más adecuado para este proyecto ha estado muy dirigida por el entorno en que se desarrolla el mismo y los datos disponibles. Según las posibles opciones para la implementación del sistema de recomendación y sus ventajas e inconvenientes, en este apartado se exponen las razones por las que se ha optado por seleccionar el algoritmo de CF basado en ítems.

En primer lugar, a la hora de tomar la decisión sobre un sistema de recomendación se ha tenido en cuenta que el número de modificaciones en los usuarios es mayor que las modificaciones en los ítems, por lo que la limitación del nuevo ítem se daría en menos casos que la del nuevo usuario. Por otro lado, al tener mayor número de usuarios que de ítems no habría limitación por falta de novedad.

Además, se ha tenido en cuenta que CF ayuda a escapar del problema de la burbuja [42]. El problema de la burbuja se refiere a la idea de que el usuario quede atrapado dentro de una burbuja, ya que con el tiempo el sistema homogeiniza tus preferencias y sólo muestra contenidos muy similares entre sí. Esto produce que el usuario vaya perdiendo interés por los resultados que el sistema ofrece.

Adicionalmente, CF ofrece cierto grado de serendipia, lo que va un paso más allá de la diversidad. La serendipia ofrece un resultado que no se está buscando pero que resulta muy valioso.

Como se mencionó anteriormente, dentro de CF se pueden encontrar dos aproximaciones: basados en usuarios y basados en ítems. Ambas aproximaciones se basan en los algoritmos k-nearest neighbours (KNN) [25]. Estos algoritmos requieren de un vecindario para realizar el cálculo de puntuaciones.

En el primer caso, el sistema de recomendación [26] le devuelve al usuario los ítems que han gustado a usuarios similares, es decir, los vecinos más cercanos al usuario a recomendar. Normalmente, el vecindario cercano lo conforman aquellos usuarios que hayan interactuado con los mismos ítems. Es habitual que se limite el número de vecinos para evitar y eliminar ruido de aquellos que no son muy similares al usuario.

En el caso del CF basado en ítems, es equivalente teniendo ítems en lugar de los usuarios. Es decir, esta técnica localiza ítems similares a los ítems guardados por el usuario.

Forma de seleccionar el algoritmo más adecuado

Para seleccionar entre ambas aproximaciones [43] [44] [45] hay que tener en cuenta cinco criterios [25]: exactitud, eficiencia, estabilidad, justificación y serendipia.

La **exactitud** de los algoritmos KNN en los métodos de recomendación depende del ratio entre el número de usuarios y el número de ítems en el sistema. La similitud entre dos usuarios en el método basado en usuarios se obtiene comparando las puntuaciones realizadas por los usuarios a un mismo

ítem. Por otro lado, en el método basado en ítems se computa la similitud entre dos ítems comparando las puntuaciones realizadas por el mismo usuario en ese ítem.

Los métodos basados en ítems producen unas recomendaciones más acertadas en casos en los que el número de usuarios es mucho mayor que el número de ítems. Por contra, los casos en los que el número de ítems es mayor que el número de usuarios se benefician más de los métodos basados en usuarios. [46] [47]

La **eficiencia** de memoria y de computación también depende del ratio entre el número de usuarios y el número de ítems. En el caso de que el número de usuarios exceda el número de ítems el método basado en ítems requiere mucha menos memoria y tiempo de computación para calcular los pesos de la similitud que los basados en usuarios.

La elección entre ambos enfoques, en cuanto a **estabilidad**, depende de la frecuencia y el número de cambios de usuarios e ítems en el sistema. Si la lista de ítems es prácticamente estática frente a la lista de usuarios, es preferible seleccionar el método basado en ítems ya que los pesos de la similitud se pueden calcular en intervalos de tiempo poco frecuentes mientras se siguen realizando recomendaciones a nuevos usuarios. Por el contrario, los métodos basados en usuarios son más estables cuando la lista de ítems disponible está en cambio constante.

Otra ventaja del enfoque basado en ítems es que se puede **justificar** de manera más sencilla una recomendación. Para explicar al usuario la recomendación que se ha realizado se le puede mostrar una lista de ítems que se han usado en la predicción. En cambio, en el enfoque basado en usuarios esta justificación resulta más complicada.

Y, por último, la **serendipia** es aquel descubrimiento valioso que se produce de manera accidental. En los métodos basados en ítems, la predicción de la puntuación de un ítem se basa en las puntuaciones dadas a ítems similares. Los sistemas de recomendación que usan este enfoque tienden a recomendar ítems que están relacionados con los gustos del usuario. Esto conlleva a recomendaciones seguras, pero no ayuda al usuario a descubrir nuevos ítems que también podrían gustarle. Por otro lado, los enfoques basados en usuarios son más propensos a realizar recomendaciones fortuitas, sobre todo si se limita a un número reducido el número de vecinos.

Resumen

Después de examinar y analizar los cinco criterios en cada uno de los enfoques, se decide optar por el método basado en ítems. El principal motivo a la hora de seleccionar este método ha sido el volumen de datos de usuarios e ítems. Movistar+ es una plataforma de televisión que cuenta con unas 25 veces más de usuarios que de contenidos, por lo que se considera que se van a obtener mejores recomendaciones con este método por todo lo descrito anteriormente.

3

Diseño

Como ya se ha comentado al comienzo de este TFM, en este proyecto se pretende realizar una aplicación para la recomendación de contenidos de la plataforma de Movistar+. El objetivo de este capítulo es explicar el diseño de cada una de las partes que lo forman, dejando lo que tiene que ver con el desarrollo para el capítulo 4.

En la Figura 3.1 se puede apreciar toda la arquitectura que se ha montado durante el TFM de manera esquemática. En el paso 0 se lleva a cabo la recopilación de las acciones del usuario, este proceso genera un fichero que se manda al servidor de recomendaciones para que calcule las recomendaciones. Como ya se ha comentado en apartados anteriores, esta monitorización sobre las acciones del usuario ha sido creada para la ejecución de este proyecto. En el paso 1, se representa la petición que realiza un usuario en su hogar desde el STB al servidor de aplicaciones. En esta petición el STB solicita los recursos necesarios de la aplicación para ejecutarla en el navegador. Después, en el paso 2, la aplicación de recomendación solicita los contenidos apropiados para dicho usuario. Luego, en el paso 3, se solicitan al servidor de metadatos la información asociada a cada uno de los contenidos que nos ha devuelvo el servidor de recomendaciones. Y, por último, se presenta al usuario la interfaz de la aplicación con los contenidos seleccionados para él.

Este capítulo se distribuye de la siguiente manera. En el apartado 3.1 se explica el servidor de recomendaciones que tiene actividad en el paso 0 de recopilación de datos y en el paso 2 a la hora de calcular las recomendaciones. Después, en el apartado 3.2 se detalla el servidor de aplicaciones. Luego, en el apartado 3.3 se expone de manera general el servidor de metadatos. Y, por último, en el apartado 3.4 se relata el diseño que se ha pensado para la aplicación.

3.1. Servidor de recomendaciones

3.1.1. Recopilación de los datos

Como se ha comentado en el capítulo anterior, para que los sistemas de recomendación puedan acertar con las predicciones que realizan a los usuarios, el sistema tiene que conocer a dicho usuario.

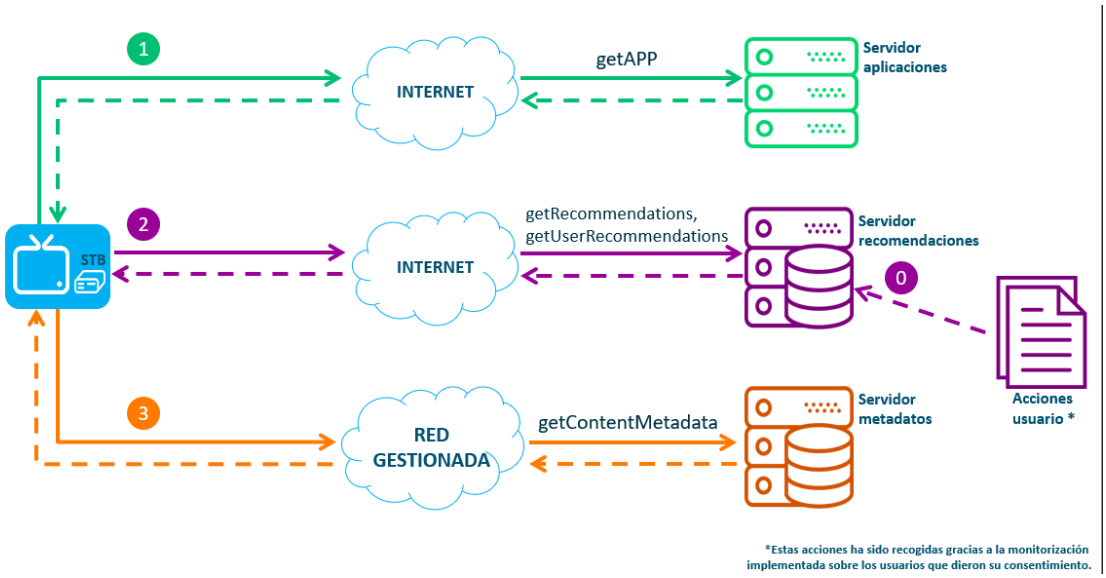

Figura 3.1: En esta figura se muestra la arquitectura que se construido para poder ofrecer las recomendaciones.

Este proyecto se centra en la actividad de visionado de un usuario y la información relacionada con la interacción de éstos con los contenidos dentro de la plataforma de Movistar+.

Los datos que se han tenido en cuenta en este proyecto se centran en las siguientes fuentes de información relacionadas con un contenido: el visionado de películas, la compra y/o alquiler, la visualización del detalle, añadir a “Favoritos” y la puntuación interna.

Hay que tener en cuenta que las valoraciones como tal no existen dentro de la plataforma, por lo que para obtener una puntuación sobre cada una de estas fuentes de información se le ha dado unos pesos diferentes teniendo en cuenta cómo funciona la interfaz y la importancia que se le quiere dar a cada fuente.

3.1.2. Arquitectura REST

Como hemos visto en el apartado 2.3, los últimos modelos hardware, vistos en las Figuras 2.2(c) y 2.2(d), tienen más capacidad que los anteriores pero aún así no se puede comparar con un equipo potente. Por eso se descarta realizar la implementación de los algoritmos de recomendación en el STB, entre otras cosas por falta de memoria y falta de procesamiento. Debido a esta restricción se propone llevar a cabo la implementación en sistemas aislados usando la tecnología REST.

REST es un tipo de arquitectura de desarrollo muy vinculado desde su origen a Hypertext Transfer Protocol (HTTP). Al igual que API, REST se puede definir como una interfaz en medio de varios sistemas que usen HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles, como XML o JSON. Surge como alternativa a otros protocolos de intercambio de datos como

Simple Object Access Protocol (SOAP), que dispone de mucha capacidad pero resulta más complejo que REST.

REST consiste en una serie de directrices o principios clave de las comunicaciones cliente-servidor. Uno de estos principios enuncia que todo lo que se intercambia en las comunicaciones es un **recurso**. Todos los recursos se representan por un formato específico que se describe por el tipo de contenido de los datos. Estos recursos pueden ser un documento, una imagen, un texto, etc. Estos recursos pueden tener **múltiples formatos**, que son diferentes de aquel en el que están almacenados. Además, manifiesta que cada recurso tiene que ser identificable por un **identificador único**, que puede ser accesible a través de una URI.

Este protocolo usa **métodos estándar** de HTTP, como GET, PUT, POST, DELETE, etc. Por ejemplo, con el método GET se obtiene información del servidor y con el método POST, por el contrario, se envía información al servidor. Estas comunicaciones que se realizan son **sin estado**, es decir, cada petición al servidor debe ser tratada de modo totalmente independiente a cualquier solicitud anterior, por lo tanto no debe haber ninguna relación entre peticiones.

Esta tecnología ha cogido mucha popularidad en los últimos años debido al incremento de los servicios web y los problemas en torno al uso de SOAP. Los servicios web se definen como aquellos servicios disponibles sobre Internet. Los servicios web que se ajustan a los principios REST reciben el nombre de RESTful.

Una de la grandes ventajas que ofrece este protocolo es la separación entre el cliente y el servidor. Este protocolo separa la interfaz que se presenta al usuario del servidor y el almacenamiento de datos. Por ejemplo, facilita la portabilidad de la interfaz a otra plataforma, permite la evolución de los distintos componentes de forma independiente, etc.

Otra de las ventajas es que API REST también es independiente del tipo de plataformas o lenguaje que se utilice. API REST siempre se adapta al tipo de sintaxis o plataforma con la que se esté trabajando en ese momento, lo que ofrece libertad a la hora de cambiar o probar nuevos entornos. Las respuestas a las peticiones que se realice siempre se tienen que hacer en el lenguaje establecido, ya sea XML o JSON.

Resumen

Como ya se ha visto, las restricciones técnicas de este proyecto han requerido instalar un servidor REST dentro de la arquitectura para servir las recomendaciones. Una de las ventajas que ofrece esta solución es la independencia. Al separar los datos del cliente, se facilita que los desarrollos se puedan realizar de manera independiente. Otra de las ventajas que destacan es la escalabilidad, ya que se puede escalar el servidor sin interferir en el cliente. En la Figura 3.1 se puede ver cómo encaja este servidor dentro de la arquitectura del proyecto.

3.2. Servidor de aplicaciones

Como su nombre indica este servidor sirve (es decir, devuelve) aplicaciones. Será llamado cada vez que un usuario solicite la aplicación de recomendación desde el STB. Este tipo de servidores, y, en particular, el que usamos en este proyecto, gestiona la lógica y el acceso a los datos de las aplicaciones que sirve.

Este servidor no se ha diseñado para el TFM y queda fuera del ámbito del mismo, pero se menciona porque forma parte de la arquitectura que se ha montado.

3.3. Servidor de metadatos

Este servidor se podría definir como un servidor de datos. Dentro de este servidor se encuentra la base de datos que contiene todos los metadatos de los contenidos que se requieren para la aplicación.

Al igual que en el apartado 3.2, este servidor no se ha diseñado para el TFM y queda fuera del ámbito del mismo, pero se menciona porque forma parte de la arquitectura que se ha montado.

3.4. Aplicación cliente

En este apartado se explica el diseño que se ha pensado para la aplicación que se presenta al usuario en su hogar. En el apartado 3.4.1 se detalla qué es un SDK y cómo funciona dentro del STB. En el apartado 3.4.2 se enseña una de las herramientas que se han desarrollado y que ha resultado muy útil en este proyecto. Para terminar, en el apartado 3.4.3 se explica el diseño que se ha seguido para la interfaz.

3.4.1. SDK

Además del API explicado en el apartado 2.2.2, se ha desarrollado un SDK. Un SDK se puede definir como un conjunto de herramientas que facilitan el desarrollo de una aplicación. Dentro de estas herramientas se encuentra la documentación de soporte, las librerías, los ejemplos de código, etc.

En este caso, el objetivo de este SDK es conseguir una segunda línea de desarrollo que permita realizar nuevas implementaciones sin interferir en la línea de desarrollo principal. Además, se ofrece en el SDK acceso a las capacidades del STB y de la plataforma, siendo agnóstico a las aplicaciones que se desarrollen. Esto permite que este entorno sea accesible por otros actores.

El desarrollo del SDK se ha realizado por fases. Primero se ha definido un API que permite la comunicación entre el core del STB y las posibles aplicaciones que se pueden desarrollar. Entre estas

llamadas podemos encontrar llamadas que permiten obtener información del cliente y del STB, gestionar el VOD y el contenido en directo y controlar el uso de las teclas del mando. Después, se han determinado los puntos de lanzamiento desde la interfaz de las posibles aplicaciones desarrolladas. Y, por último, se han agrupado las llamadas del API para gestionar los permisos de acceso al core de cada grupo. En la Figura 3.2 se muestra el módulo SDK dentro de la arquitectura general del STB. Como se puede observar, tanto la aplicación del desarrollo principal como el módulo del SDK se comunican directamente con el navegador del sistema operativo. Y, también, se pueden comunicar entre ellos como por ejemplo para solicitar información del usuario o informar de algún evento.

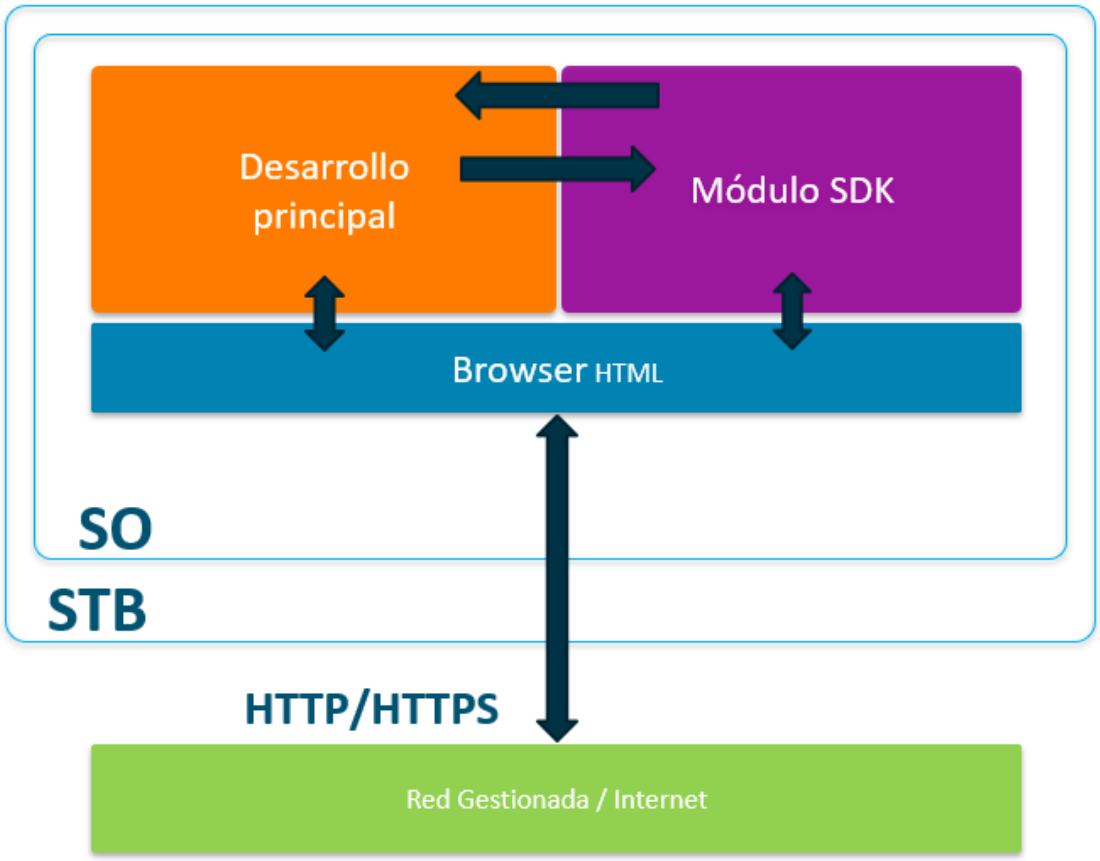

Figura 3.2: Figura que muestra la arquitectura del Software Development Kit dentro del Set-Top Box.

3.4.2. Simulador

Dentro del SDK, también se ha desarrollado una herramienta que permite simular un STB en un navegador cualquiera. En esta herramienta se pueden seleccionar diversas opciones, entre ellas: el tamaño de la TV, el punto de lanzamiento de la aplicación y limitar el tipo de acceso de la aplicación. Esta herramienta ha sido muy útil en este proyecto para no depender de un STB. Las pruebas finales siempre se han realizado sobre el STB porque la potencia de éste es mucho menor que la del portátil donde se ha realizado el desarrollo, por lo tanto se ha querido probar en un entorno de ejecución real.

En la Figura 3.3 se puede ver una captura de esta herramienta.

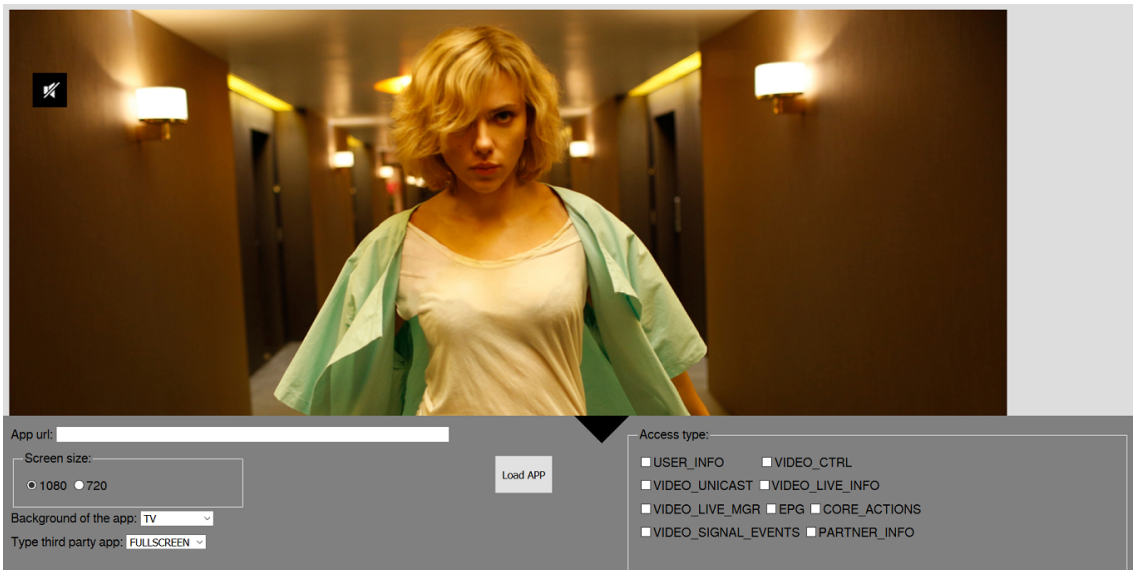

Figura 3.3: Figura que muestra una captura de la herramienta que permite simular el STB.

3.4.3. Interfaz de la aplicación

Como se ha explicado en el apartado 3.4.1, uno de los motivos para lanzar el SDK es añadir una línea de desarrollo secundaria sin interferir con la línea de desarrollo principal. Esta línea de desarrollo secundaria se ejecuta dentro del módulo del SDK. La aplicación principal de este TFM se ha desarrollado siguiendo las reglas básicas de estilo del resto de la plataforma, facilitando así que el usuario no pierda el contexto en el que se encuentra. Además, se ha intentado mostrar la información de manera clara y ordenada al usuario.

El diseño de la experiencia de usuario se basa en unas reglas básicas que ayudan a los desarrolladores a entender a los usuarios potenciales de la aplicación. Un ejemplo de estas reglas básicas son "Laws of UX" [48] que proporcionan una colección de principios máximos que los desarrolladores pueden considerar cuando desarrollan interfaces, entre las cuales se encuentran las siguientes:

1.– **Efecto de usabilidad estética**: los usuarios perciben que un diseño estético agradable es un diseño más usable. Estos diseños pueden hacer a los usuarios más tolerantes a los errores menores.

2.– **Ley de Fitts**: diseñar elementos que sean fáciles de seleccionar y posicionarlos cerca del usuario.

3.– **Ley de Hick**: el tiempo que tarda un usuario en tomar una decisión aumenta a medida que se incrementa el número de opciones.

4.– **Ley de Jakob**: a los usuarios les gusta conocer cómo funcionan las interfaces, por eso

tiende a seguir unos patrones de diseño.

5.– **Ley de similitud**: el ojo humano tiende a percibir elementos similares de un diseño como una imagen completa.

6.– **Ley de Miller**: Las personas pueden recodar una media de siete elementos en su memoria. Para evitar la confusión y la pérdida de usuario se mantienen conjuntos de elementos menores que siete.

Dentro de la plataforma existe una cabecera común a todas las páginas de navegación. Desde esta cabecera se puede encontrar el título de la página en la que nos encontramos, a parte de la fecha, la hora y el logo de Movistar+. Gracias a esta cabecera el usuario es consciente del tiempo en ese momento y de dónde se encuentra.

Después se encuentra el resto de la página principal, que, como se verá más adelante, ha sufrido modificaciones respecto a lo diseñado inicialmente. Estas modificaciones se realizaron para incorporar los comentarios que hicieron los usuarios durante las pruebas.

A continuación se representan de manera esquemática los prototipos de la interfaz. En la primera Figura 3.4 se puede ver que nada más terminar la cabecera se muestran los contenidos. Estos contenidos que se presentan son los que devuelve el recomendador para el usuario. Esto causaba una sensación de agobio por parte de los usuarios al ver tantos contenidos de golpe. Por lo que se ha optado por seguir la estructura de la página principal de Movistar+. Como se muestra en la Figura 3.5, antes de mostrar los contenidos se presenta un carrusel con imágenes grandes y luego los contenidos.

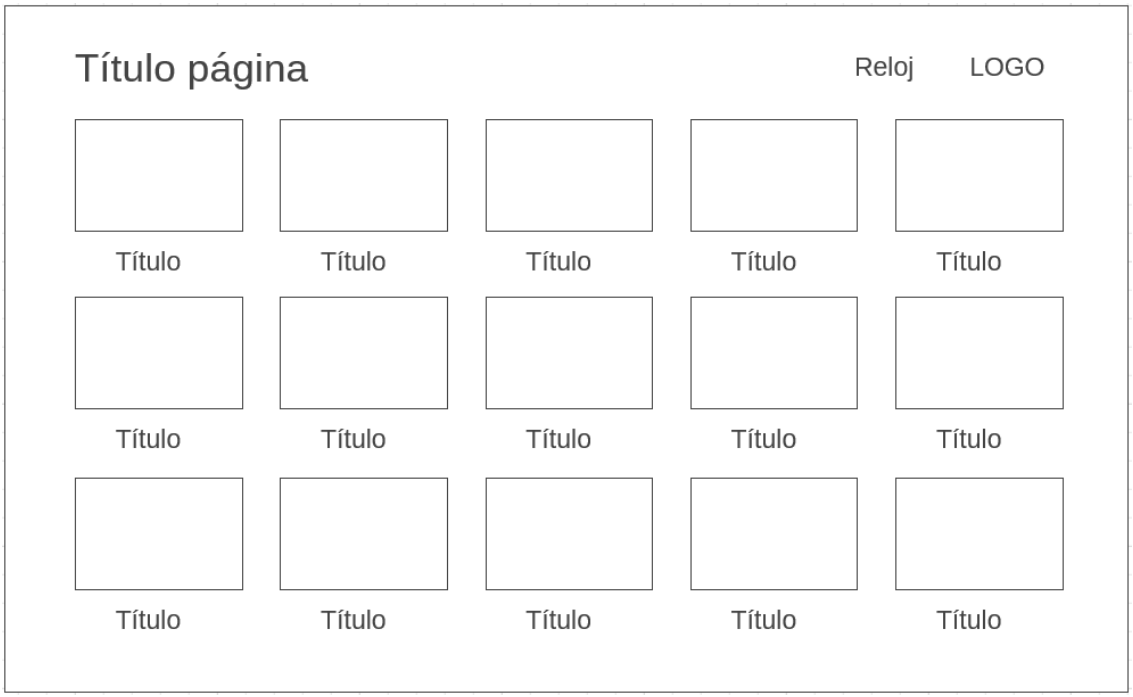

Figura 3.4: Figura que muestra el primer prototipo de la aplicación

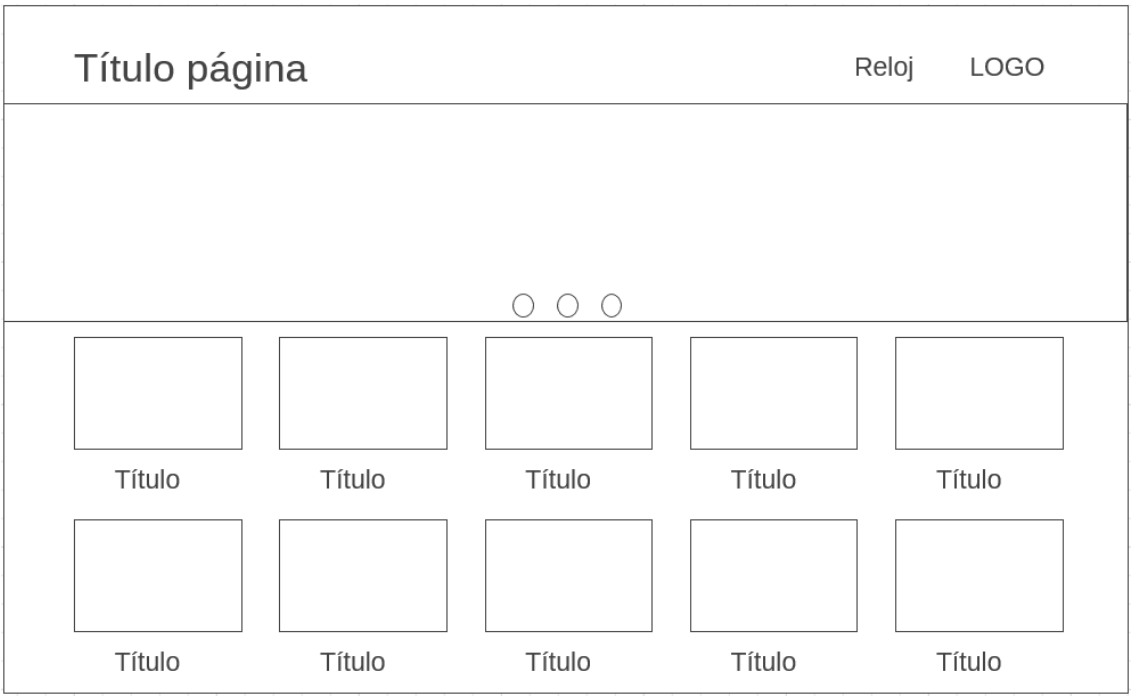

Figura 3.5: Figura que muestra el prototipo final de la aplicación

4

Desarrollo

El objetivo de este capítulo es explicar cómo se ha implementado cada uno de los tres objetivos enunciados en la introducción de este trabajo. Hay que notar que todo el desarrollo del proyecto se ha realizado con un equipo portátil LG gran 15 con un procesador Intel Core i5. Así mismo se ha empleado una máquina virtual con un Ubuntu Server de 64 bits.

Este capítulo se distribuye de la siguiente manera. En el apartado 4.1 se explican los desarrollos realizados en el servidor de recomendaciones. Después, en el apartado 4.2, se detalla el servidor de aplicaciones. Luego, en el apartado 4.3, se exponen de manera general el servidor de metadatos. Y, por último, en el apartado 4.4 se describe el desarrollo llevado a cabo para la aplicación de recomendación.

4.1. Servidor de recomendaciones

Este servidor es un Ubuntu Server de 64 bits que se ha montado en una máquina virtual. En este servidor se producen varias operaciones, por un lado, tal y como se explica en el apartado 4.1.1, es responsable de procesar los datos extraídos de la plataforma y almacenar la salida en un fichero. Por otro lado, se calculan las recomendaciones una vez obtenido el fichero con los datos esperados utilizando la librería descrita en el apartado 4.1.2. Y, por último, este servidor sirve estas recomendaciones calculadas mediante la librería detallada en el apartado 4.1.3.

Como comentario general de este apartado todos los datos con los que se ha trabajado han sido obtenidos de la monitorización implementada a los usuarios que dieron su consentimiento para la realización de esta prueba de concepto.

4.1.1. Extracción de los datos

Dentro de la plataforma de Movistar+ se recoge información de las acciones del usuario como por ejemplo: los contenidos que se visualizan, los canales que visitan, la navegación que realizan, las secciones a las que acceden, los literales de búsqueda, etc. Esta información se procesa en varios sistemas y, finalmente, se vuelca en diversos ficheros con una estructura determinada.

A pesar del procesamiento y el filtrado que se hace previamente en la plataforma, estos ficheros pueden llegar a ser muy pesados, pudiendo superar los 15GB por fichero. Por eso se ha utilizado **AWK** que es un lenguaje de programación muy potente para la gestión de ficheros.

Después de reflexionar, se ha decidido dividir el proceso en 3 partes, que son: filtrado, procesamiento y normalización.

Filtrado de los datos

En este paso se revisan los ficheros generados por las acciones del usuario, se filtran para quedarnos con los datos que son útiles y se unifican en un sólo fichero. En nuestro caso, como ya se dijo en el capítulo anterior, los ficheros que interesan son los que tienen los siguientes datos: el visionado de películas, la compra y/o alquiler de contenido, la visualización del detalle de un contenido, añadir a "Favoritos" un contenido y la puntuación interna que tenga cada ítem.

Cada uno de estos ficheros tiene una estructura propia pero en todos se encuentran los siguientes campos, entre otros: identificador del usuario dentro de la plataforma, marca de tiempo con la fecha y la hora, dispositivo desde el que se lanza, acción que se realiza, identificador del contenido sobre el que se realiza la acción, etc.

Al terminar este paso se generan dos ficheros: uno con las acciones del usuario y otro con todos los contenidos disponibles en la plataforma. Los ficheros tienen la siguiente estructura:

- **ID USUARIO:** Identificador del usuario dentro de la plataforma.
- **ACCIÓN REALIZADA:** Acción que realiza el usuario sobre un contenido.
- **ID CONTENIDO:** Identificador del contenido sobre el que se realiza la acción.

ID USUARIO | *...* | *ACCIÓN REALIZADA* | *ID CONTENIDO*

Cuadro 4.1: Esta es la estructura del primer fichero que se genera.

- **ID CONTENIDO:** Identificador del usuario dentro de la plataforma.
- **DURACIÓN CONTENIDO:** Duración del contenido en milisegundos. Este dato se utilizará para controlar que el usuario ha visualizado más del 80 % del contenido. Se ha determinado que la visualización del 80 % del contenido es un valor suficiente como para asegurar que el usuario ha visualizado el contenido por completo, ya que es necesario dejar un margen trasero que no incluya las pantallas de créditos.
- **PUNTUACIÓN INTERNA:** Puntuación que tiene el contenido internamente dentro de la plataforma. Dentro de la plataforma este dato se utiliza para dar visibilidad a ciertos contenidos sobre otros. En nuestro caso, se utilizará para desempatar entre dos contenidos después de

ejecutar el algoritmo.

ID CONTENIDO | DURACIÓN CONTENIDO | PUNTUACIÓN INTERNA

Cuadro 4.2: Esta es la estructura del segundo fichero que se genera.

Procesamiento de los ficheros

Una vez terminado el filtrado de los ficheros se comienzan a procesar esos dos ficheros obtenidos. En este paso se procede a puntuar cada una de las acciones del usuario. Después de realizar unas pruebas con diferentes pesos se ha decidido seleccionar los siguientes:

Acción	Puntuación
Marcado como favorito	1.5
Visualización contenido	1.25
Visualización detalle	1
Alquiler contenido	1

Tabla 4.1: Pesos de las acciones del usuario.

Luego se suman todas las interacciones sobre el mismo usuario y el mismo contenido para obtener un fichero con la siguiente estructura:

- **ID USUARIO:** Identificador del usuario dentro de la plataforma.
- **ID CONTENIDO:** Identificador del contenido sobre el que se realiza la acción.
- **PUNTUACIÓN OBTENIDA:** Suma total de la puntuación obtenida sobre el usuario y el contenido, ponderando la puntuación de cada acción por su peso asociado.

ID USUARIO | ID CONTENIDO | PUNTUACIÓN OBTENIDA

Cuadro 4.3: Esta es la estructura del tercer fichero que se genera.

Normalización del fichero

Para terminar, se normalizan las puntuaciones obtenidas en el último fichero, con un valor máximo de cinco y un valor mínimo de uno. La estructura de este último fichero es similar al del anterior:

- **ID USUARIO:** Identificador del usuario dentro de la plataforma.
- **ID CONTENIDO:** Identificador del contenido sobre el que se realiza la acción.

- **PUNTUACIÓN NORMALIZADA:** Suma total de la puntuación obtenida sobre el usuario y el contenido después de normalizarla.

ID USUARIO	*ID CONTENIDO*	*PUNTUACIÓN NORMALIZADA*

Cuadro 4.4: Esta es la estructura del ultimo fichero que se genera.

Para realizar la normalización primero se tienen que establecer unos valores mínimos y máximos para los nuevos valores. En nuestro caso, se ha seleccionado 1 como valor mínimo y 5 como valor máximo. En la siguiente ecuación 4.1 se puede ver la fórmula que se ha seguido en este proceso, donde $o = \{o_i\}$ es el conjunto sobre el que se trabaja y $n = \{n_i\}$ es el nuevo conjunto que surge al realizar esta transformación.

$$n_i = f(o_i) = \frac{\text{máx}_i(n_i) - \text{mín}_i(n_i)}{\text{máx}_i(o_i) - \text{mín}_i(o_i)} * ((o_i - \max_i(o_i)) + \max_i(n_i)) = \frac{5-1}{0-20} * ((o_i - 20) + 5) \quad (4.1)$$

Después de ejecutar este último paso, obteniendo todos los valores entre el valor mínimo de 1 y un valor máximo de 5, se eliminan todos los ficheros temporales que se han creado en pasos anteriores. De esta manera, se mantiene limpia la carpeta de salida y no hay confusión a la hora de seleccionar el fichero correcto.

Alternativa estudiada

Se ha valorado la opción de almacenar los datos obtenidos en una base de datos como HBase [49], MongoDB [50], Cassandra [51], etc. Finalmente se descartó esta alternativa para este proyecto porque, a parte de instalar, configurar y mantener la base de datos, después de realizar unas pequeñas pruebas con RankSys se comprobó que los tiempos de calcular las recomendaciones en el momento de solicitarlas eran asumibles para esta prueba de concepto.

Además, se tendría que definir un API para que la aplicación se comunicara con el sistema de recomendación y que éste sirviera las sugerencias, lo cual habría retrasado mucho el resto de etapas del proyecto.

4.1.2. Librería de recomendación

En este apartado se presentan las librerías estudiadas para la recomendación de contenidos. Después, se exponen los motivos que llevaron a la selección de una sobre otra.

RankSys

RankSys [52] es una librería Java para la implementación y evaluación de algoritmos y técnicas de recomendación. Esta librería esta programada en Java 8 y aprovecha la características que esta versión ofrece como el uso de funciones lambda, Stream y las facilidades para la paralelización automática de código.

La versión disponible es la 0.4.3 que incluye la implementación de varios algoritmos de CF, así como una amplia variedad de métricas de novedad y diversidad. Los módulos publicados en esta versión son los siguientes:

- **RankSys-core**: clases comunes y auxiliares de las librerías.
- **RankSys-fast**: soporte de implementaciones rápidas y eficientes de la estructura de datos y los algoritmos.
- **RankSys-metrics**: interfaces y componentes comunes para la definición de las métricas.
- **RankSys-rec**: soporte para la generación de las listas de recomendación.
- **RankSys-nn**: algoritmo de recomendación de vecinos cercanos.
- **RankSys-mf**: algoritmo de recomendación de factorización de matrices.
- **RankSys-fm**: máquinas de factorización para recomendación usando JavaFM.
- **RankSys-lda**: algoritmos de filtrado colaborativo Latent Dirichlet Allocation.
- **RankSys-novdiv**: recursos comunes para las métricas de novedad y diversidad.
- **RankSys-novelty**: métricas de novedad y técnicas de mejora.
- **RankSys-diversity**: métricas de diversidad y técnicas de mejora.
- **RankSys-compression**: estado del arte de las técnicas de compresión para el filtrado colaborativo basado en memoria.
- **RankSys-examples**: ejemplos de uso de los módulos anteriores.

El desarrollo de este apartado se ha basado en los códigos que se facilitan en el módulo de ejemplos (RankSys-examples). En esos códigos de ejemplo se recogen cuatro argumentos: la ruta del fichero de usuarios, la ruta del fichero de contenidos, la ruta del fichero de entrenamiento y la ruta del fichero de test. Después, se ejecutan hasta diez algoritmos de recomendación. Se obtiene un fichero de salida por cada una de esas ejecuciones. Todos los ficheros tienen la estructura que se muestra en el cuadro 4.5, que se parece a la obtenida en el fichero de normalización:

ID USUARIO | *ID CONTENIDO* | *PUNTUACIÓN OBTENIDA*

Cuadro 4.5: Esta es la estructura general de los ficheros de salida.

Para evitar el empate entre dos contenidos recomendados a un mismo usuario, se ha utilizado la puntuación interna que se tiene del contenido. En la clase **SimpleRecommendationFormat** se construye la tupla de usuario, ítem y valor, por lo que se ha modificado esta tupla para multiplicar ese valor por la puntuación interna.

A este código se le ha añadido la evaluación de los algoritmos. Por un lado se encuentran las métricas individuales y por otro lado las métricas del sistema. Las métricas individuales se componen de:

- **Precision**: proporción de ítems relevantes en una lista de recomendación.
- **Recall**: proporción de ítems relevantes en una lista de recomendación en función de todos los ítems relevantes.
- **Normalized Discounted Cumulative Gain**: es una métrica que mide la calidad del ránking.
- **Expected Popularity Complement**: es una métrica de novedad que mide el número esperado de ítems recomendados relevantes que no se hayan visto previamente (es decir, novedosos).

Las métricas del sistema se componen de:

- **Aggregate diversity**: contabiliza el número total de elementos que recomienda el sistema. Es una medida relevante para evaluar en qué medida un inventario está expuesto a los usuarios [52].
- **Gini Index sales diversity**: representa la distribución del descubrimiento de un ítem.
- **User Coverage**: ratio de usuarios para los que el sistema puede proporcionar una recomendación [52].

Con estas métricas se han podido realizar diferentes pruebas modificando las variables que vienen por defecto en la librería con el objetivo de obtener el mejor resultado. En el apartado 5.1.1 se detallan las pruebas que se han llevado a cabo.

Alternativas estudiadas

En este caso se han considerado tres alternativas: LensKit [53], MyMediaLite [54] y Surprise [55]. En primer lugar, LensKit es un conjunto de herramientas, tanto en Java como en Python, que proveen soporte para entrenar, ejecutar y evaluar algoritmos de recomendación de una manera flexible y adecuada para la investigación y la educación. La versión en Python se basa en el proyecto sucesor basado en Java, por lo que se encuentra más actualizada que la de Java. En segundo lugar, MyMediaLite es una biblioteca de sistemas de recomendación para .NET. Esta librería aborda los dos escenarios más comunes en el CF: la evaluación de la predicción y la predicción de ítems sobre acciones positivas de los usuarios. Por último, Surprise es una librería de Python para analizar sistemas de

recomendación. Esta librería ha sido diseñada para ofrecer a los usuarios un control sobre los experimentos que realicen, por eso tratan de redactar una documentación clara y precisa. En esta librería se proporcionan varios algoritmos ya implementados listos para usar, como por ejemplo vecinos cercanos y factorización de matrices. Además, se proveen herramientas para evaluar, analizar y comparar el rendimiento de los algoritmos.

La librería de MyMediaLite se descartó por desconocimiento del lenguaje en el que estaba programado. Entre las dos opciones de Python se profundizó en la última de ellas [56] [57]. Se instaló la librería de Surprise en el servidor y se realizaron una serie de pruebas sobre los ficheros obtenidos. Desafortunadamente, no se consiguieron los resultados esperados ya que los tiempos de ejecución eran mayores que los de RankSys. Por ese motivo, se descartó esta librería.

4.1.3. REST4RecSys

Este apartado detalla la librería utilizada para montar el servidor REST. Esta librería [58] se ha utilizado para desarrollar la comunicación entre la aplicación y el servidor donde se calculan las recomendaciones. El objetivo de esta librería es servir la lista de recomendaciones de un usuario. Esta librería se ha instalado en el servidor de la máquina virtual y se ha modificado para las necesidades del proyecto. REST4RecSys levanta un servicio en el puerto 8080 y se mantiene escuchando hasta que un servicio web le manda una petición.

Para este proyecto se han añadido dos llamadas GET. Por un lado, se ha implementado la llamada **getRecommendations** que obtiene las recomendaciones populares de la plataforma. En esta petición se solicita un número máximo de contenidos, ya que en el carrusel se mostrarán sólo tres. Por otro lado, la llamada **getUserRecommendations** calcula la lista de recomendaciones del usuario indicado en la petición. Al contrario que la llamada anterior, que era una llamada genérica, en esta se especifica un identificador de usuario.

Como se puede ver en el código más adelante, la aplicación realiza una petición GET con un identificador de usuario. El servidor procesa la petición para ese usuario concreto y devuelve un JSON con las recomendaciones de ese usuario. En caso de que el usuario no exista o no se tenga información sobre él, se devuelve el listado de los contenidos más populares de la plataforma.

Código 4.1: Código de ejemplo de una de las llamadas desarrolladas.

```
@GET
@Path("/user/get/{uid}/recommendations")
@Produces(MediaType.APPLICATION_JSON)
```

4.2. Servidor de aplicaciones

Este servidor es necesario para servir los recursos necesarios de la aplicación de recomendación cuando se solicitan desde el STB o desde el simulador, explicado en el apartado 3.4.2.

En este servidor no se ha desarrollado nada específico para el TFM, por lo que queda fuera de su ámbito. Se menciona porque forma parte de la arquitectura necesaria para el proyecto.

4.3. Servidor de metadatos

En la aplicación desarrollada se presenta al usuario ciertos contenidos. Para enriquecer estos contenidos, y no mostrar únicamente el ID, es necesario hacer uso de este servidor. En este servidor se encuentra la información relacionada con el ID del contenido que nos devuelve el servidor de recomendación. Esta información puede ser el título, la carátula, la duración, la sinopsis, el género, etc.

No se ha desarrollado nada nuevo sobre este servidor, pero se creía conveniente explicar de forma general por formar parte de la arquitectura del proyecto.

4.4. Aplicación cliente

En esta aplicación se le muestra al usuario los tres contenidos más populares de la plataforma en modo carrusel y, además, sus recomendaciones personalizadas en forma de listado. Como se ha comentado en el apartado 1.2 estas aplicaciones que corren dentro del STB se lanzan en un navegador, por este motivo la aplicación se ha desarrollado en HTML, con la ayuda de CSS y JS.

Se ha desarrollado una página web dinámica que se genera en función de las acciones que vaya realizando el usuario. Como se verá más adelante en el documento, esto se realizó entre otras cosas para no sobrecargar la memoria del STB. Esta aplicación se comunica con diferentes sistemas como se puede ver en el diagrama 3.1.

La aplicación se construye con dos **librerías**, un **script** y una **hoja de estilos**. Entre las librerías se encuentra la **librería** que se desarrolló en el SDK y una **librería** para realizar la depuración de la aplicación. Esta última librería permite depurar la aplicación hasta terminar con el desarrollo, por lo que en el momento de subir a producción esta librería se podría descartar. Junto a estas dos librerías, se ubican el script y la hoja de estilos que se ha desarrollado durante el proyecto.

El **script** desarrollado controla diferentes funciones necesarias en la aplicación. En primer lugar, se suscribe a las teclas del mando a distancia necesarias en la aplicación. Si no se realiza esta suscripción la aplicación desarrollada no recibiría ningún evento ante la pulsación de cualquier tecla y por tanto no

se ejecutaría la funcionalidad asociada para esa tecla. Una vez se ha realizado el alta de las teclas, se procede a gestionar las funcionalidades de cada una de ellas. En este caso, se utilizan las flechas para mover el curso por la aplicación y el botón “Ok” para acceder al detalle del contenido.

Después, se controlan las conexiones que se realizan a los diferentes sistemas, el servidor de recomendación y la base de datos que almacena los metadatos. La primera comunicación que se realiza al iniciar la aplicación es la obtención de los contenidos recomendados para dicho usuario. A la que el servidor contesta con una lista de identificadores de contenido. Luego, se solicita a la base de datos los metadatos asociados a cada uno de los contenidos. Esta solicitud de metadatos requiere una petición por cada contenido que haya en la lista de recomendación. En un principio, estas llamadas se realizaban secuencialmente, pero finalmente se modificó para que se realizaran en paralelo. Se tuvo especial cuidado con la gestión de la respuesta de estas llamadas porque el listado está ordenado de manera descendente y el servidor puede no responder en el mismo orden en el que se realiza la petición. Además, ambas llamadas cuentan con un timeout definido en caso de que el sistema no responda. Si salta este timeout en el caso de la petición de recomendación se muestra al usuario un mensaje de error y no le permite acceder a la aplicación. En cambio, si salta en la obtención de metadatos se elimina ese contenido del listado y se muestra el resto.

En este script, también se manejan los contenidos que se muestran al usuario. Para ello se van liberando los contenidos que no se muestren en función de la posición del usuario, tanto por arriba como por abajo. Gracias a esto no se sobrecarga la memoria del navegador y el STB puede ir más fluido.

La **hoja de estilos** mantiene el estilo del resto de la plataforma, de tal manera que el usuario no note el cambio. Dentro del estilo que se ha mantenido se encuentra por ejemplo la cabecera, el tipo de letra, tamaño de las carátulas, borde de la selección, espaciado entre filas, etc. Como se ha explicado en el apartado 3.4 en un primer momento se desarrolló la aplicación listando únicamente los contenidos recomendados por el usuario, pero después de recibir el feedback de los testers se decidió separar la visualización de los contenidos en dos módulos. Por un lado, las recomendaciones populares de la plataforma y, por otro, las recomendaciones personalizadas del usuario. El primer módulo se basa en el algoritmo de popularidad y para el segundo módulo se utiliza el CF basado en ítems. En la Figura 4.1 se muestra la interfaz definitiva, después de realizar los cambios comentados.

Resumen

Gracias al SDK se ha podido desarrollar desde el portátil, pudiendo hacer pruebas en el simulador con más libertad y facilidad, ya que no se dependía de un STB. Gracias al simulador se ha podido depurar toda la lógica de la aplicación eliminando todos los fallos antes de realizar las pruebas en el STB. Para comprobar la velocidad de respuesta, la aplicación se tiene que examinar directamente en el STB. Como ya se ha comentado, en el apartado 2.3, la capacidad de procesamiento, de cálculo, de

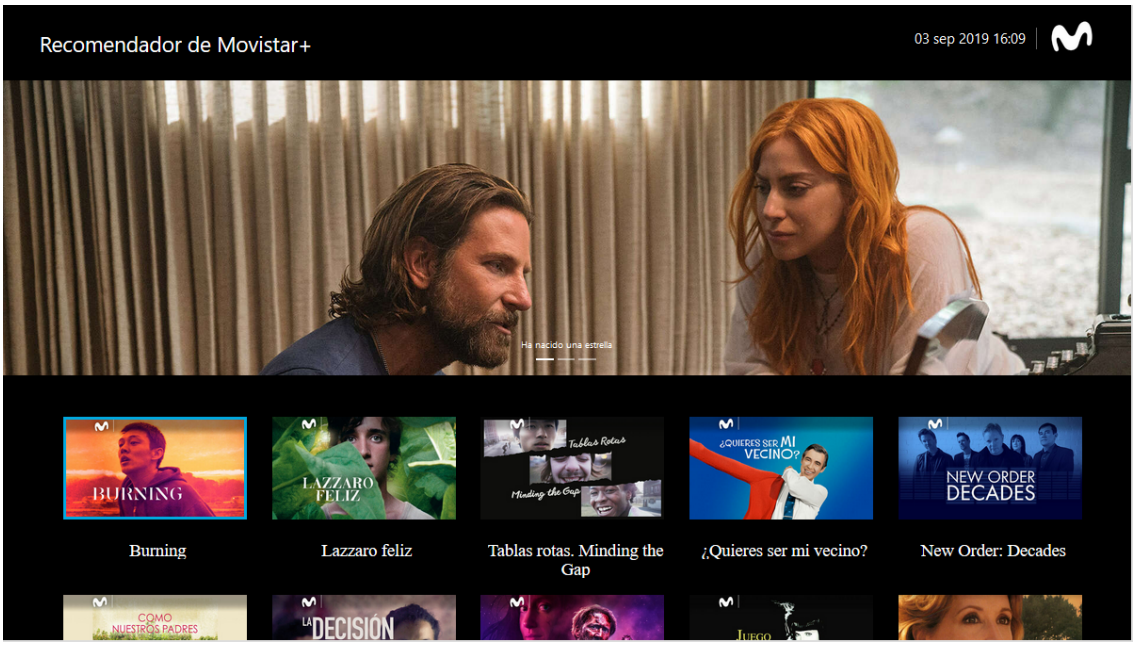

Figura 4.1: Figura que muestra el desarrollo final de la aplicación.

memoria, etc. del STB es inferior a la capacidad que tiene el portátil donde se ha desarrollado. Por este motivo, es importante confirmar el correcto funcionamiento en el dispositivo hardware donde se va a ejecutar en el entorno real.

5

Integración, pruebas y resultados

Una vez finalizado el desarrollo se procede a realizar todas las pruebas que sean necesarias para verificar el correcto funcionamiento de la aplicación. Estas pruebas permiten detectar y corregir errores cometidos a la hora de implementar la aplicación.

Al usar una metodología de desarrollo basada en Scrum [12], en todo momento se dispone de un desarrollo en funcionamiento pero con escasa funcionalidad. A medida que pasa el tiempo se va aumentando la funcionalidad manteniendo la parte del ciclo anterior.

En cuanto se tiene una parte de la funcionalidad se realizan pruebas en un ámbito simulado (servidores simulados, llamadas simuladas, etc) encontrando así todos los fallos posibles antes de pasarlo a un entorno real. De esta manera se irá depurando el desarrollo a medida que se avanza el proyecto.

5.1. Pruebas unitarias

En este apartado se van a detallar las pruebas que se realizaron en cada uno de los módulos que se ha desarrollado en este proyecto.

En el apartado 5.1.1 se explican tanto las pruebas que se llevaron a cabo en cuanto a la recogida de los datos como las que se ejecutaron en la libería de RankSys. Después, el apartado 5.1.2 expone alguno de los resultados obtenidos de la encuesta llevada a cabo después del desarrollo.

5.1.1. Servidor de recomendaciones

En el servidor de recomendaciones se han llevado a cabo pruebas respecto a las dos líneas de desarrollo. Por un lado, se encuentran las pruebas relativas a la extracción de los datos de la plataforma y, por otro lado, las pruebas de la librería de RankSys utilizada.

Extracción de los datos

Como se ha comentado en el apartado 4.1.1, este script trabaja con ficheros muy pesados, por lo que se requiere mucho tiempo para procesar y ejecutar todos los pasos. Uno de los objetivos principales es reducir el tiempo de ejecución.

La primera versión del script desarrollada en BASH tardaba una media de 20 minutos. Al tener que recorrer varios ficheros la solución pasaba por bucles anidados, lo que aumenta el tiempo de cálculo. A continuación se muestra un ejemplo del código en BASH.

Código 5.1: Código de ejemplo del desarrollo en BASH.

```
while IFS='|' read -r ID UCID_A
do
        while IFS='|' read -r UCID_C DUR RAT
        do
                if [ "$UCID_A" == "$UCID_C" ]
                then
                        echo $ID","$UCID_A","$RAT >> tmp.txt
                fi
        done < $FICHERO_CMS_S
done < $FICHERO_TMP
```

El tiempo de ejecución con la solución BASH se consideraba alto por lo que se invirtió en cambiar el desarrollo de BASH por AWK. Uno de los grandes cambios que se realiza es la eliminación del bucle anidado, lo que ayudó a reducir el tiempo de ejecución aproximadamente en un 50 %.

Código 5.2: Código de ejemplo del desarrollo en AWK.

```
awk -F"|" '{
    if(NR == FNR){
      CMS[$1]++
    }else{
                if($2 in CMS){
                        print $0
                }
    }
}' $FICHERO_TMP $FICHERO_CMS_S > tmp.txt
```

En la Figura 5.1 se muestra la ejecución del script con la implementación final. Con esta implementación se consiguió que la ejecución del script con ficheros de más de un millón de líneas se mantuviera por debajo de los 10 minutos. La ejecución de este script se realiza offline para tener el fichero listo cuando se levante el servidor REST que se ha montado en el servidor de recomendaciones.

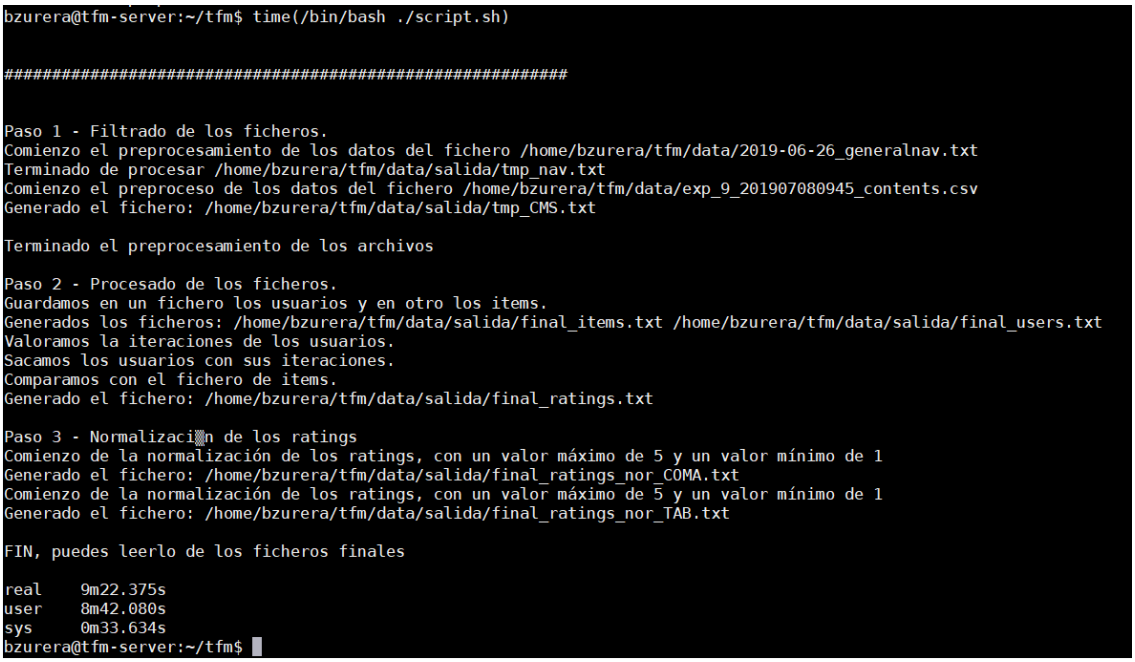

Figura 5.1: Figura que muestra el tiempo que tarda en ejecutar el script con todos los pasos.

Librería de recomendación

Con RankSys se hicieron algunas pruebas para evaluar los diferentes algoritmos. Como se ha comentado en apartado 4.1.2 se añadió código para evaluar cada uno de los diez algoritmos; no obstante, las pruebas se centraron sobre todo en el algoritmo CF.

Los datos que se utilizaron para las pruebas están detallados en la Tabla 5.1. Después, en las siguientes tablas se puede ver un extracto de los resultados obtenidos en las pruebas que se realizaron sobre los algoritmos. Concretamente, en la Tabla 5.2 se pueden comprobar los resultados obtenidos de las métricas individuales. Mientras que en la Tabla 5.3 se encuentran los resultados obtenidos de las métricas del sistema.

Prueba	**UB**		**IB**	
	k	**q**	**k**	**q**
1	100	1	10	1
2	1000	1	100	1
3	100	10	10	10
4	2000	1	200	1
5	2000	10	200	10
6	5000	1	500	1

Tabla 5.1: Esta tabla refleja los datos que se han utilizado en las pruebas realizadas.

Algoritmo	Precision	Recall	NDCG	EPC
pop	0,0534	0,2358	0,1631	0,9413
rnd	0,0011	0,0033	0,0020	0,9988
IB, k=10, q=1	0,0158	0,0560	0,0322	0,9840
IB, k=100, q=1	0,0193	0,0698	0,0388	0,9806
IB, k=10, q=10	0,0147	0,0513	0,0284	0,9851
IB, k=200, q=1	0,0194	0,0705	0,0390	0,9803
IB, k=200, q=10	0,0147	0,0514	0,0284	0,9851
IB, k=500, q=1	0,0195	0,0709	0,0392	0,9803
UB, k=100, q=1	0,0223	0,0884	0,0579	0,9753
UB, k=1000, q=1	0,0386	0,1547	0,1025	0,9604
UB, k=100, q=10	0,0197	0,0775	0,0498	0,9782
UB, k=2000, q=1	0,0418	0,1684	0,1106	0,9573
UB, k=200, q=10	0,0311	0,1270	0,0807	0,9681
UB, k=5000, q=1	0,0439	0,1763	0,1178	0,9552

Tabla 5.2: Esta tabla muestra los resultados obtenidos de las métricas individuales.

Algoritmo	AggrDiv	UserCvg	GINI
pop	19,0000	61764.0	0,0032
rnd	2,9080	61764.0	0,9642
IB, k=10, q=1	26,7153	23248.0	0,2147
IB, k=100, q=1	2,6660	23248.0	0,1973
IB, k=10, q=10	2,6715	23248.0	0,2214
IB, k=200, q=1	2,6630	23248.0	0,1941
IB, k=200, q=10	26,7194	23248.0	0,2214
IB, k=500, q=1	26,5906	23248.0	0,1912
UB, k=100, q=1	22,9819	20608.0	0,0996
UB, k=1000, q=1	15,2862	23248.0	0,0406
UB, k=100, q=10	24,0507	20608.0	0,1228
UB, k=2000, q=1	14,2003	23248.0	0,0314
UB, k=200, q=10	2,1147	23248.0	0,0875
UB, k=5000, q=1	0,1378	23248.0	0,0254

Tabla 5.3: Esta tabla presenta los resultados obtenidos de las métricas del sistema.

5.1.2. Aplicación

Las versiones de la aplicación que se fueron liberando a lo largo del desarrollo se fueron instalando en el entorno de preproducción. Se considera preproducción un entorno previo a la puesta en producción. Los desarrollos que se instalan en este entorno son desarrollos con las funcionalidades implementadas, pero sin la certificación completa, por lo que pueden salir errores.

En nuestro caso, ha sido muy útil ir liberando la aplicación, sobre todo, para depurar la aplicación, probarlo en un entorno completamente real sin simuladores y, también, poder obtener el feedback de los usuarios beta. De hecho, se ha realizado una encuesta sencilla con Microsoft Forms sobre estos usuarios y se han obtenido 48 respuestas que han resultado muy útiles para mejorar la interfaz. Los resultados de esta encuesta se pueden encontrar en el apartado 5.3.

5.2. Pruebas de integración

Estas pruebas consisten en verificar que un conjunto de partes software que funcionan de manera separada, tal y como se ha comprobado en las pruebas unitarias, también lo hacen de manera conjunta. En esta fase los módulos individuales son combinados y probados como grupo.

Para realizar estas pruebas se ejecutará la aplicación mientras el servidor que sirve las recomendaciones está levantado. Se pone especial atención a los tiempos de respuesta de dicho servidor de recomendaciones.

Mediante la Figura 3.1 vista previamente se muestran los distintos componentes necesarios para el correcto funcionamiento de la aplicación y las relaciones existentes entre ellos de forma simplificada.

En el diagrama se muestra cómo el dispositivo IPTV conectado en el hogar del usuario en primer lugar solicita la aplicación al servidor de aplicaciones. Después, es capaz de comunicarse con el servidor a través de Internet para obtener las recomendaciones personalizadas para el usuario. Finalmente, una vez obtenidas las recomendaciones se comunica con otro servidor interno para solicitar los metadatos de los contenidos y poder mostrárselo al usuario.

Como se ha visto en el apartado apartado 4.1.3 se hace uso de la librería REST4RecSys [58] para implementar la comunicación entre la aplicación y el servidor. Este servidor se encuentra ubicado en un dominio diferente al de la aplicación, por lo que es necesario utilizar las cabeceras Cross Origin Resource Sharing (CORS). CORS [59] es un mecanismo que utiliza cabeceras HTTP adicionales para permitir que un user agent obtenga permiso para acceder a recursos seleccionados desde un servidor, en un origen distinto (dominio) al que pertenece. Un agente crea una petición HTTP de origen cruzado cuando solicita un recurso desde un dominio distinto, un protocolo o un puerto diferente al del documento que lo generó.

Por razones de seguridad, los exploradores restringen las solicitudes HTTP de origen cruzado iniciadas dentro de un script. CORS da controles de acceso a dominios cruzados para servidores web y transferencia segura de datos en dominios cruzados entre navegadores y servidores web. Los exploradores modernos utilizan CORS en un contenedor API (como XMLHttpRequest o Fetch) para ayudar a mitigar los riesgos de solicitudes HTTP de origen cruzado. En esta aplicación se utiliza **XMLHttpRequest** [60].

Para habilitar estas cabeceras se configura el servidor REST4RecSys [58]. En el siguiente código se puede ver el código necesario para la activación de dichas cabeceras en el servidor Java que se ha modificado.

Código 5.3: Código de ejemplo para la activación de las cabeceras.

```
Response respuesta = Response.ok(list)
    .header("Access-Control-Allow-Origin", "*")
    .header("Access-Control-Allow-Credentials", "true")
    .header("Access-Control-Allow-Headers", "Content-type, Content-Type")
    .build();
```

Si no se configuran estas cabeceras, la respuesta se recibe correctamente, de hecho se recibe un "200 OK", ya que la conexión se realiza correctamente pero no se tiene acceso al contenido de la respuesta. Como se ha explicado es una medida de seguridad incorporada en la mayoría de los navegadores para evitar acceder a dominios cruzados. En las Figuras 5.2 y 5.3 se pueden comprobar las diferencias de las respuestas al configurar las cabeceras CORS en el servidor web. Concretamente, en la Figura 5.2 se puede ver un ejemplo de la respuesta que se recibe si no se configuran las cabeceras CORS. Por el contrario, en la Figura 5.3 se observa un ejemplo en el se configuran las cabeceras adecuadamente y se puede comprobar cómo efectivamente se puede acceder al contenido JSON de la respuesta. Se remarcan en rojo las diferencias con la anterior, que serían las cabeceras configuradas y posibilidad de acceder al JSON obtenido en la respuesta. En ambas figuras se han eliminado datos confidenciales, como la IP o el ID del usuario.

Además, durante estas pruebas se identificaron algunos problemas, por lo que se realizaron modificaciones en el script de la aplicación. Primero se confirmó que las llamadas de los metadatos no mantenían la sesión activada en el servidor y el BE tenía que comprobar la autenticidad del usuario con cada una de las llamadas. La solución a este problema se ve a continuación en la siguiente sección. Luego se comprobó que la aplicación realizaba llamadas innecesarias que podían poner en peligro el BE de la plataforma, esto se explica en la última sección de este apartado.

INICIO SESIÓN

La plataforma tiene varios mecanismos para protegerse contra usuarios no legítimos. Uno de estos mecanismos consiste en comprobar la autenticación del usuario que realiza la llamada. Lo que su-

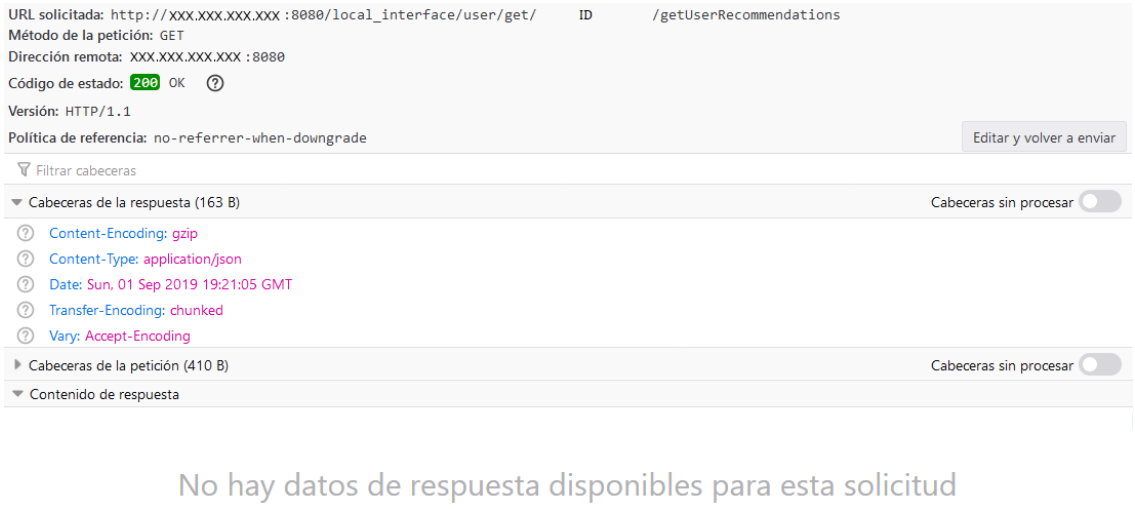

Figura 5.2: En esta figura se muestra una captura de la conexión sin cabeceras.

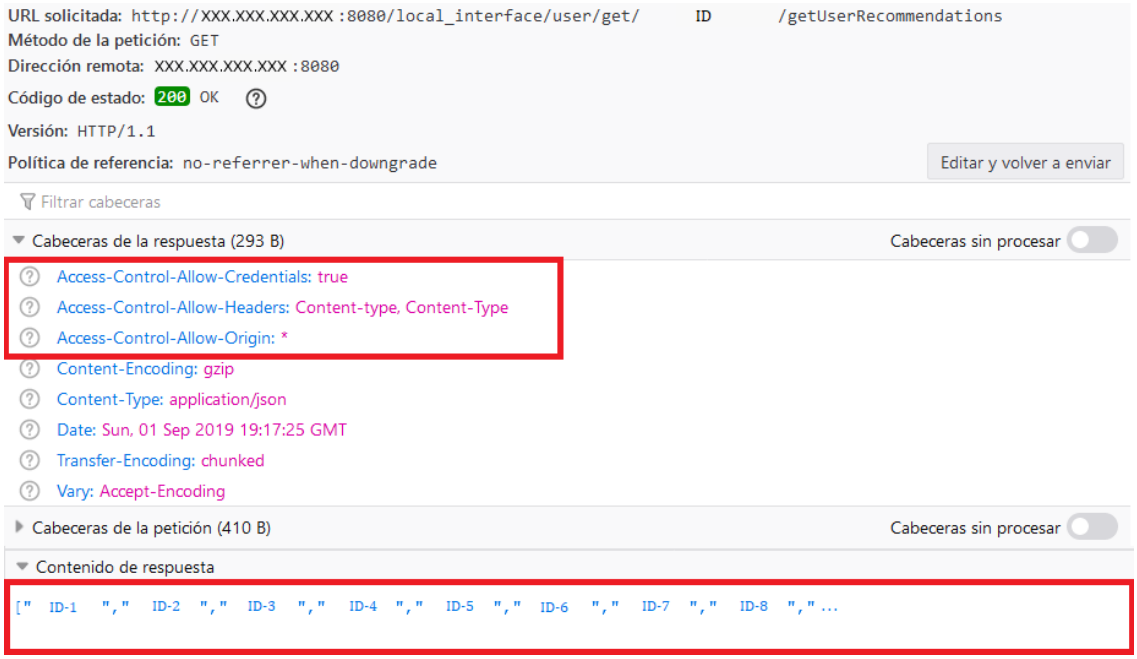

Figura 5.3: En esta figura se muestra una captura de la conexión con cabeceras.

pone peticiones intermedias adicionales a la base de datos de los usuarios. Estas peticiones no son inmediatas porque tienen que atravesar varios sistemas, lo que puede ralentizar la respuesta.

Para ello se hace uso de una de las propiedades de las cabeceras CORS [59]. La propiedad **XMLHttpRequest.withCredentials** [61] es un booleano que indica si las peticiones CORS se tienen que hacer con credenciales como cookies, cabeceras de autorización o certificados TLS. Por defecto, en las invocaciones XMLHttpRequest [60] de un sitio cruzado, los navegadores no envían estas credenciales.

Código 5.4: Código de ejemplo para activar la propiedad de las credenciales.

```
var xhr = new XMLHttpRequest();
xhr.open('GET', 'http://example.com/', true);
xhr.withCredentials = true;
xhr.send(null);
```

La primera vez que el servidor reciba una llamada con las credenciales comprobará la autenticidad de la petición. En las próximas llamadas al servidor recibidas con las mismas credenciales, y gracias a esta implementación, el servidor no necesita volver a verificar la autenticidad de la petición. Gracias a esta solución se reducen considerablemente las solicitudes intermedias, por lo que se consigue que la respuesta sea más rápida.

PAGINACIÓN

Al integrar todos los componentes se vió que la llamada de **getUserRecommendations**, explicada en el apartado 4.1.3, devolvía una lista con más de 2500 ítems. Con esta lista la aplicación pide los metadatos al BE de la plataforma. Esta petición se realiza ítem por ítem, es decir, que tendremos tantas peticiones como ítems haya. Esto puede provocar una avalancha de peticiones innecesarias al BE, ya que los usuarios normalmente seleccionan un contenido dentro de las primeras posibilidades que se le muestran.

Esto se puede solucionar de varias maneras. La primera que se planteó era limitar el número de ítems que devuelve la llamada **getUserRecommendations**. Esta solución es una buena aproximación pero no es completa ya que se siguen realizando peticiones innecesarias. Como se ha comentado, normalmente los usuarios sólo visualizan las primeras filas de contenidos, por lo que aunque la respuesta se limite a 100 contenidos se siguen haciendo peticiones innecesarias.

Finalmente, la solución por la que se optó pasa por desarrollar un mecanismo de protección en la aplicación. Este mecanismo consiste en paginar las peticiones e ir realizándolas en función de la posición en que se encuentre el usuario. Según vaya bajando en los contenidos, se van realizando las peticiones. De esta manera se evitan las peticiones innecesarias al BE.

Esta variable de paginación es configurable, por lo que se puede modificar en caso de ser nece-

sario. La variable toma por defecto el valor de 15. El cálculo se ha realizado en función de la interfaz. En la primera pantalla se muestra el carrusel y se visualizan dos filas de contenidos, lo que hacen una suma de 10 contenidos en total. De esta manera se tiene un margen, en caso de que el usuario sea muy rápido.

5.3. Resultados

Para evaluar la aplicación realizada durante este TFM se ha realizado una encuesta a una serie de usuarios beta que son usuarios avanzados de la plataforma. Esta encuesta ha sido respondida por 48 personas y en este apartado se encontrarán los resultados obtenidos. En la encuesta se han realizado siete preguntas, que son:

1.– ¿Te ha gustado la aplicación?

2.– ¿Te han resultado útiles las recomendaciones que ha realizado la aplicación?

3.– ¿Te ha parecido sencillo el uso de la aplicación?

4.– ¿Qué es lo que más te ha gustado?

5.– ¿Qué es lo que menos?

6.– ¿Te gustaría que hubiera una aplicación de este tipo dentro de la plataforma?

7.– Si pudieras añadirle alguna funcionalidad, ¿cuál sería?

A continuación, se detallan alguno de los resultados. Por un lado, se van a evaluar los resultados sobre las recomendaciones y, por otro lado, se analizarán los resultados obtenidos sobre la interfaz de usuario. Como se puede ver en la Figura 5.4 los usuarios han encontrado útiles los contenidos que le ha devuelto la aplicación, ya que un 65 % de los usuarios ha votado 4 o más.

Se ha averiguado que esos 3 usuarios que votaron por debajo de 2 exponían que todos los contenidos que le recomendaba la aplicación eran contenidos infantiles, que era lo que visualizaban sus hijos. Por lo que solicitan una mejora para distinguir al usuario que está utilizando el STB en ese momento. Para llevar a cabo esta mejora habría que utilizar un sistema híbrido, tal y como se explicó en el apartado 2.4.3.

Respecto a la interfaz de usuario, se pueden sacar dos conclusiones. La primera conclusión, y como se puede ver la Figura 5.5, de las 48 personas que respondieron la encuesta a 44 (es decir, un 92 %) les gustaría tener una aplicación de este tipo dentro de la plataforma. Además, la mayoría de esas 44 personas han dado una puntuación alta (por encima de cuatro) para la pregunta 1.

Analizando en profundidad sobre esas cuatro personas que han contestado que no, todas coinciden en que no se necesita una aplicación adicional a parte. Consideran que es necesario un algoritmo de recomendación, pero preferirían que se integrara dentro de la plataforma que ya existe.

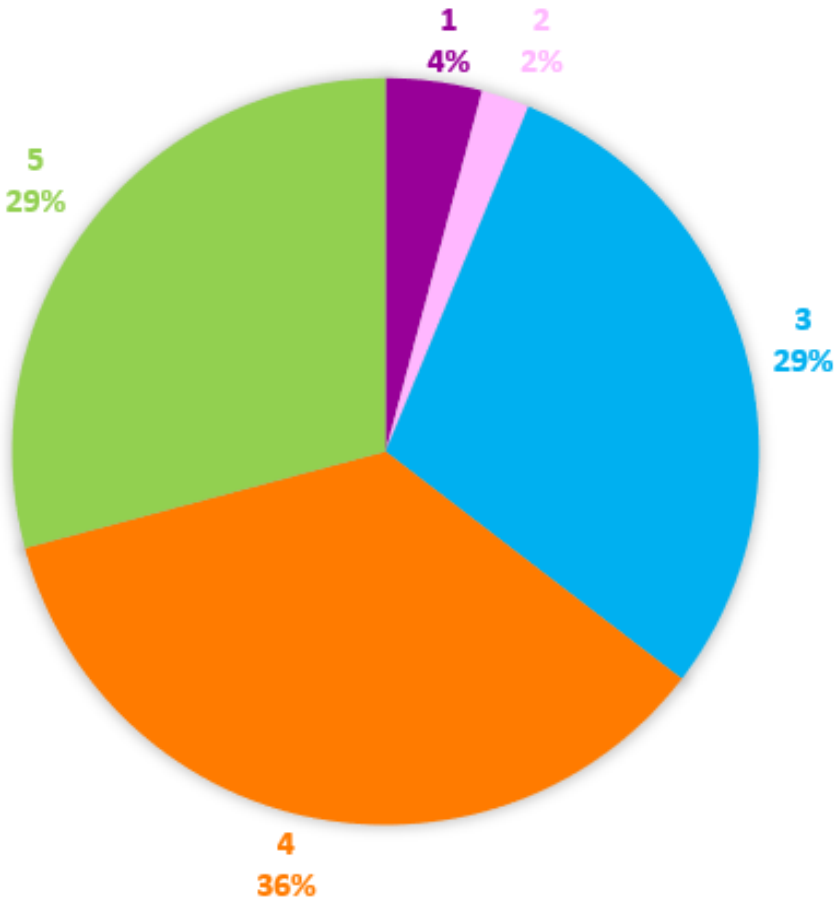

Figura 5.4: Figura que muestra el resultado de la pregunta número 2.

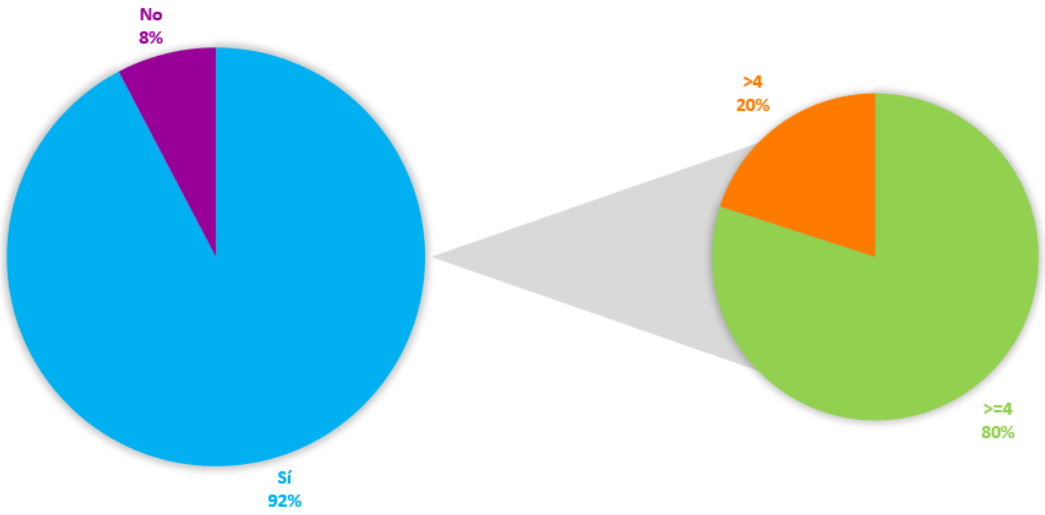

Figura 5.5: Figura que muestra el resultado de la pregunta número 6 y 1.

La segunda conclusión se apoya en la Figura 5.6 que se muestra los resultados obtenidos de la pregunta 3. Se consideran muy buenos, ya que casi un 85 % de los usuarios ha encontrado la aplicación sencilla de usar. Esto ha sido gracias a mantener y adaptar a nuestra aplicación el estilo utilizado en el resto de la plataforma.

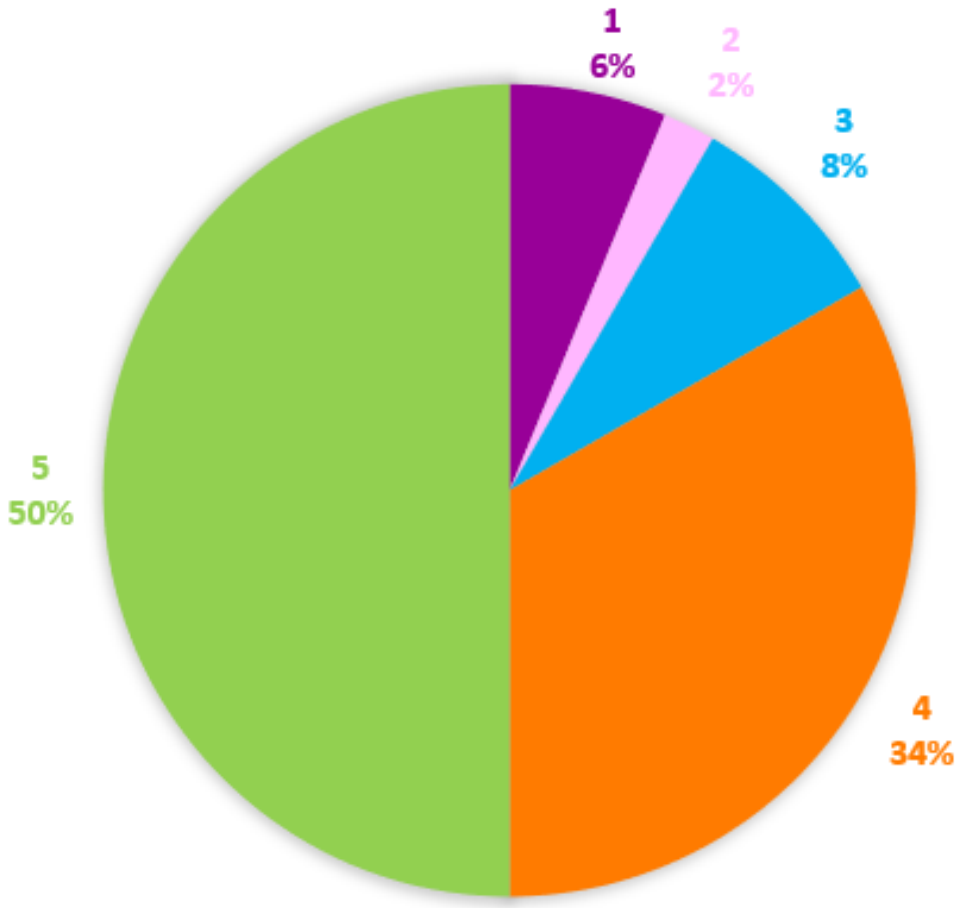

Figura 5.6: Figura que muestra el resultado de la pregunta número 3.

Dentro de las preguntas libres se han encontrado diversidad de respuestas. Una de las respuestas más repetidas que se priorizó era que los usuarios encontraban agobiante la cantidad de contenidos que se mostraba nada más entrar en la aplicación. Por este motivo, se modificó la aplicación para incorporar un carrusel y así separar los contenidos en dos módulos. Por un lado, se muestran los contenidos más populares, gracias a la llamada **getRecommendations**, y por otro, los contenidos personalizados para el usuario gracias a la llamada **getUserRecommendations**, ambas explicadas en el apartado 4.1.3.

En general, el proyecto ha **resultado muy positivo** como se han podido ver en los resultados de la encuesta. Se puede deducir que los usuarios encuestados están a favor de implementar un algoritmo de recomendación en esta plataforma, como la prueba de concepto desarrollada durante este TFM.

6

Conclusiones y trabajo futuro

En este capítulo se va a explicar en primer lugar las conclusiones a las que se han llegado durante la realización de este proyecto. Después, se van a enunciar las líneas de trabajo futuras que se podrán llevar a cabo.

6.1. Conclusiones

En este proyecto se ha desarrollado una aplicación para la recomendación de contenidos de la plataforma de Movistar+. En primer lugar se ha diseñado todo un SDK dentro del dispositivo hardware utilizado en la plataforma de Movistar+. Este SDK permite trabajar en varias líneas de desarrollo en paralelo sin interferir en la línea de desarrollo principal. Después se han analizado los sistemas de recomendación que existen actualmente para optar por el sistema más adecuado para el proyecto. La selección del sistema de recomendación ha estado muy ligada por el entorno en el que se desarrolla el proyecto y los datos disponibles. Una vez analizadas las opciones junto con sus ventajas e inconvenientes, se ha optado por seleccionar el algoritmo de CF basado en ítems. Por último, se ha desarrollado la interfaz de la aplicación web desde donde el usuario podrá acceder a los contenidos recomendados que devuelve dicho algoritmo.

Gracias al trabajo de varios meses, se ha obtenido como resultado una aplicación web completamente funcional y con un diseño amigable para el usuario. Para hacer posible que un usuario real pudiera lanzar la aplicación desde su hogar y obtener los contenidos recomendados ha sido necesario montar una arquitectura tipo REST, tal y como se puede en la Figura 3.1. A lo largo de la memoria se explica cómo ha sido todo el proceso. Gracias al diseño elegido, esta aplicación se podido probar en un entorno real y con usuarios reales que se han ofrecido a participar en la prueba de concepto. Durante las pruebas ha quedado claro que los sistemas de recomendación son útiles para guiar al usuario en la elección de un contenido para visualizar. Actualmente, el número de elecciones posibles en la plataforma es enorme y, gracias a técnicas como los sistemas de recomendación, el usuario no necesita buscar entre miles de contenidos para seleccionar lo que desea ver.

La realización de este proyecto me ha permitido reforzar y aplicar todos los conocimientos adquiri-

dos tanto a lo largo de los cursos del grado y máster como los obtenidos en mi etapa profesional. Me ha permitido ampliar mis conocimientos en el lenguaje de programación Java, el desarrollo de interfaces gráficas para las aplicaciones web y, sobre todo, en los sistemas de recomendación.

6.2. Trabajo futuro

Esta aplicación desarrollada a lo largo de estos meses puede ser mejorada y ampliada en varios aspectos. Desde el punto de vista del servidor de recomendaciones:

- **Automatización del proceso de obtención de datos**: en este proyecto la extracción de los datos se ha realizado de manera manual mediante la ejecución de un script, por lo que se plantea unir el sistema de recomendaciones con el sistema de datos. De esta manera se evitarían las ejecuciones manuales.
- **Implementación basada en sistemas híbridos**: Con esta aproximación se podrían enriquecer las recomendaciones del usuario teniendo en cuenta el contexto en el que se encuentra a la hora de realizar la petición.

Desde el punto de vista de la interfaz de usuario se han agrupado cuatro ideas principales, gracias a los resultados obtenidos en las encuestas:

- **Filtrado de contenidos**: los usuarios expresaron la necesidad de poder filtrar los contenidos que le presenta la aplicación. Este filtro podría activarse mediante una tecla del mando y se podría dar la facilidad de filtrar según ciertos criterios, como el género, la clasificación por edad, año de la publicación, etc.
- **Información adicional**: a los usuarios les gustaría que la aplicación ofreciera información sobre el contenidos que se muestra. Actualmente se muestra el título y la carátula, por lo que se propone que en las próximas versiones se incorpore un pop up con información adicional del contenido seleccionado.
- **Justificación de la recomendación**: de la misma manera que Netflix o Spotify explican el por qué de sus recomendaciones, se plantea incorporar esa justificación en la aplicación.

Bibliografía

[1] "Netflix España - Ver series en línea, ver películas en línea." https://www.netflix.com/es/. Accessed 20 August 2019.

[2] "Ver las Mejores Series Online | HBO España." https://es.hboespana.com/. Accessed 20 August 2019.

[3] "Amazon.es: Prime Video." https://www.amazon.es/b?ie=UTF8&node=12092245031. Accessed 20 August 2019.

[4] "YouTube." https://www.youtube.com/. Accessed 20 August 2019.

[5] "Apple TV." https://www.apple.com/es/tv/. Accessed 21 August 2019.

[6] "Disney Channel | El canal oficial de las series y los juegos de Disney Channel." https://tv.disney.es. Accessed 21 August 2019.

[7] ATRESPLAYER, "ATRESPLAYER | Series online, programas, noticias, deportes y TV en directo." https://www.atresplayer.com/, Aug. 2019. Accessed 21 August 2019.

[8] "TV online y series a la carta - Mitele.es - Mitele.es." https://www.mitele.es/. Accessed 21 August 2019.

[9] Ericsson, "ConsumerLab report on TV and Media 2017 - Ericsson," tech. rep., Ericsson Consumer LAB, 2017. Accessed 20 June 2019.

[10] AIMC, "Marco General de los medios de España - 2018," tech. rep., Asociación para la Investigación de Medios de Comunicación, 2018. Accessed 19 June 2019.

[11] Ericsson, "Ericsson Mobility Report June 2019," tech. rep., Ericsson, 2019. Accessed 01 July 2019.

[12] "SCRUM." https://www.scrum.org/index. Accessed 21 August 2019.

[13] "What is Scrum?." https://www.scrum.org/resources/what-is-scrum. Accessed 22 August 2019.

[14] "Google Hangouts." https://hangouts.google.com/. Accessed 25 August 2019.

[15] "What is an API?." https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces. Accessed 28 August 2019.

[16] "Conozca más sobre la tecnología Java." https://www.java.com/es/about/. Accessed 25 August 2019.

[17] B. Zurera, "Gestor de trabajo de fin de grado y/o máster," bachelor's thesis, Universidad de Burgos, 2015. Supervisor: Carlos López.

[18] "Home | Dropwizard." https://www.dropwizard.io/1.3.14/docs/. Accessed 01 August 2019.

[19] "HTML Standard." https://html.spec.whatwg.org/multipage/. Accessed 24 August 2019.

[20] "Sass: Syntactically Awesome Style Sheets." https://sass-lang.com/. Accessed 18 August 2019.

[21] "Getting started | Less.js." http://lesscss.org/. Accessed 18 August 2019.

[22] "JavaScript." https://developer.mozilla.org/es/docs/Web/JavaScript. Accessed 24 August 2019.

[23] "A Brief History of Set-Top Box Innovation." https://blogs.cisco.com/sp/a_brief_history_of_set-top_box_innovation, Mar. 2010. Accessed 17 August 2019.

[24] R. Baldwin, "The History of the Set-Top Box: From Bunny Ears to TV," *Wired*, Mar. 2013. Accessed 17 August 2019.

[25] F. Ricci, L. Rokach, and B. Shapira, eds., *Recommender Systems Handbook*. Springer, 2015.

[26] A. Bellogín, *Recommender System Performance Evaluation and Prediction: An Information Retrieval Perspective*. PhD thesis, Universidad Autónoma de Madrid, 2012. Supervisors: Pablo Castells and Iván Cantador.

[27] A. Bellogín, "Performance prediction in recommender systems: application to the dynamic optimisation of aggregative methods," Master's thesis, Universidad Autónoma de Madrid, 2009. Supervisor: Pablo Castells.

[28] S. Berkovsky, I. Cantador, and D. Tikk, eds., *Collaborative Recommendations - Algorithms, Practical Challenges and Applications*. WorldScientific, 2018.

[29] Artificial Intelligence - All in One, "Lecture 43 — Collaborative Filtering | Stanford University." https://www.youtube.com/watch?v=h9gpufJFF-0. Accessed 2 July 2019.

[30] R. Chen, Q. Hua, Y. Chang, B. Wang, L. Zhang, and X. Kong, "A Survey of Collaborative Filtering-Based Recommender Systems: From Traditional Methods to Hybrid Methods Based on Social Networks," *IEEE Access*, vol. 6, pp. 64301–64320, 2018.

[31] P. G. Campos, A. Bellogín, I. Cantador, and F. Díez, "Time-Aware Evaluation of Methods for Identifying Active Household Members in Recommender Systems.," in *CAEPIA* (C. Bielza, A. Salmerón, A. Alonso-Betanzos, J. I. Hidalgo, L. Martínez-López, A. T. Lora, E. Corchado, and J. M. Corchado, eds.), vol. 8109 of *Lecture Notes in Computer Science*, pp. 22–31, Springer, 2013.

[32] A. Bellogín, "Performance prediction in recommender systems," in *User Modeling, Adaption and Personalization - 19th International Conference, UMAP 2011, Girona, Spain, July 11-15, 2011. Proceedings* (J. A. Konstan, R. Conejo, J. Marzo, and N. Oliver, eds.), vol. 6787 of *Lecture Notes in Computer Science*, pp. 401–404, Springer, 2011.

[33] G. Adomavicius and A. Tuzhilin, "Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, pp. 734–749, June 2005.

[34] M. Burgess, "This is how Netflix's secret recommendation system works," *Wired UK*, Aug. 2018.

[35] R. Shankar, "Build Your own Recommendation Engine-Netflix Demystified:Demo+Code." https://towardsdatascience.com/build-your-own-recommendation-engine-netflix-demystified-demo-code-550401d4885e,

Dec. 2018. Accessed 1 July 2019.

[36] A. Yu, "How Netflix Uses AI and Machine Learning." https://becominghuman.ai/how-netflix-uses-ai-and-machine-learning-a087614630fe, Feb. 2019. Accessed 01 July 2019.

[37] "Netflix Has A Plan To Change The Way You Chill." https://www.buzzfeednews.com/article/nicolenguyen/netflix-recommendation-algorithm-explained-binge-wat Accessed 01 July 2019.

[38] J. Blattmann, "Netflix: Binging on the Algorithm." https://uxplanet.org/netflix-binging-on-the-algorithm-a3a74a6c1f59, Aug. 2018. Accessed 01 July 2019.

[39] A. Pasick, "The magic that makes Spotify's Discover Weekly playlists so damn good." https://qz.com/571007/the-magic-that-makes-spotifys-discover-weekly-playlists-so- Accessed 01 July 2019.

[40] "Amazon's recommendation secret." https://fortune.com/2012/07/30/amazons-recommendation-secret/. Accessed 01 July 2019.

[41] "Amazon's Recommendation Engine: The Secret To Selling More Online." http://rejoiner.com/resources/amazon-recommendations-secret-selling-online/. Accessed 01 July 2019.

[42] E. Pariser, *The Filter Bubble: What the Internet Is Hiding from You*. Penguin Group , The, 2011.

[43] P. Grover, "Various Implementations of Collaborative Filtering - Towards Data Science." https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0, Dec. 2018. Accessed 14 July 2019.

[44] "Collaborative Filtering RecSys: User-User and Item-Item - Recommender Systems." https://es.coursera.org/lecture/machine-learning-applications-big-data/collaborative-filtering-recsys-user-user-and-item-item-X7tMz. Accessed 2 July 2019.

[45] B. Wang, "Comparison of User-Based and Item-Based Collaborative Filtering." https://medium.com/@wwwbbb8510/comparison-of-user-based-and-item-based-collaborative-filtering-f58a1c8a3 Aug. 2018. Accessed 2 July 2019.

[46] F. Fouss, A. Pirotte, J. Renders, and M. Saerens, "Random-Walk Computation of Similarities between Nodes of a Graph with Application to Collaborative Recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, pp. 355–369, Mar. 2007.

[47] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001* (V. Y. Shen, N. Saito, M. R. Lyu, and M. E. Zurko, eds.), pp. 285–295, ACM, 2001.

[48] "Aesthetic Usability Effect | Laws of UX." https://lawsofux.com/. Accessed 02 August 2019.

[49] "Apache HBase – Apache HBase™ Home." https://hbase.apache.org/. Accessed 28 July 2019.

[50] "Open Source Document Database." https://www.mongodb.com. Accessed 28 July 2019.

[51] "Apache Cassandra." http://cassandra.apache.org/. Accessed 28 July 2019.

[52] S. Vargas, *Novelty and diversity enhancement and evaluation in Recommender Systems*. PhD thesis, Universidad Autónoma de Madrid, 2015. Supervisor: Pablo Castells.

[53] "LensKit." https://lenskit.org/. Accessed 14 July 2019.

[54] "MyMediaLite Recommender System Library." http://www.mymedialite.net/. Accessed 14 July 2019.

[55] N. Hug, "Suprise." http://surpriselib.com/. Accessed 14 July 2019.

[56] Ola, "Recommender System made easy with Scikit-Surprise - Hacktive Devs." https://medium.com/hacktive-devs/recommender-system-made-easy-with-scikit-surprise-569cbb689824, Mar. 2019. Accessed 14 July 2019.

[57] S. Li, "Building and Testing Recommender Systems With Surprise, Step-By-Step." https://towardsdatascience.com/building-and-testing-recommender-systems-with-surprise-step-by-step-d4ba702ef80b, Apr. 2019. Accessed 11 July 2019.

[58] A. Bellogín, "Code for ACM RecSys 2018 demo "Towards an open, collaborative REST API for Recommender Systems": abellogin/REST4recsys." https://github.com/abellogin/REST4RecSys, Nov. 2018. Accessed 29 July 2019.

[59] "Control de acceso HTTP (CORS)." https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS. Accessed 03 August 2019.

[60] "XMLHttpRequest." https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest. Accessed 04 August 2019.

[61] "XMLHttpRequest.withCredentials." https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/withCredentials. Accessed 04 August 2019.

Definiciones

contenido bajo demanda Los contenidos bajo demanda, conocido por sus siglas en inglés Video On-Demand, son aquellos cotenidos en los que el usuario decide cuándo visualizarlos.

filtrado colaborativo El filtrado colaborativo utiliza la puntuación de otros usuarios e ítems en el sistema. La idea principal es que, teniendo dos usuarios similares, es decir, que han puntuado de manera similar en el pasado, el sistema realizará recomendaciones al segundo usuario basándose en las puntuaciones que haya dado el primer usuario a aquellos ítems desconocidos para el segundo.

IPTV Televisión sobre protocolo IP. Se utiliza una red de distribución privada, por lo que se puede garantizar la calidad de imagen y sonido. Se puede encontrar más información en el 2.2.1.

ítem Se llama ítem a los objetos que se recomiendan. El valor de un ítem puede ser positivo si es útil para el usuario, o negativo en caso contrario.

navegador Aplicación que, mediante enlaces de hipertexto, permite navegar por una red informática.

OTT Basada en una conexión a internet de acceso público, compartiendo la red con el resto de tráfico. Se puede encontrar más información en el 2.2.1.

REST Es un tipo de arquitectura de desarrollo muy vinculado desde su origen a HTTP. Se puede leer más información en el apartado 3.1.2.

SDK Es conjunto de herramientas software que facilitan al desarrollador la creación de una nueva aplicación. Para más detalle revisar el apartado 3.4.1.

STB Receptor de señales digitales que permite reproducir estas señales en la televisión. Es un dispositivo receptor o decodificador de las señales (analógicas o digitales) de televisión analógica o digital (DTV), para luego ser mostrada o visualizada en el televisor (u otro dispositivo de televisión). Se puede encontrar más información en el apartado 2.3.

usuario beta Es un grupo de usuarios a los que se les presenta un producto de manera anticipada, es decir, aquellos productos de los que no se ha terminado la fase de desarrollo y pueden presentar fallos. Estos usuarios probaran el producto en un entorno real.

Acrónimos

API Application Programming Interface.

BE Back-End.

CBF Content-based filtering.

CF Collaborative Filtering.

CORS Cross Origin Resource Sharing.

CSS Cascading Style Sheets.

HF Hybrid Filtering.

HTML HyperText Markup Language.

HTTP Hypertext Transfer Protocol.

IP Internet Protocol.

IPTV Internet Protocol Television.

JS JavaScript.

KNN k-nearest neighbours.

OTT Over-The-Top.

REST Representational State Transfer.

SDK Software Development Kit.

SOAP Simple Object Access Protocol.

STB Set-Top Box.

TDT Television Digital Terrestre.

TFM Trabajo Fin de Máster.

TV televisión.

VOD Video On-Demand.

Apéndices

A

Plan del proyecto

La planificación temporal de este proyecto se ha realizado utilizando las metodologías ágiles. Por lo que tendremos una serie de iteraciones completas en un tiempo impuesto por los componentes del proyecto.

Al comienzo de cada iteración se definen una serie de tareas a completar durante esa iteración. De esta manera resultará más fácil adaptarse a los nuevos cambios o los problemas que hayan podido surgir durante el desarrollo de la aplicación. A cada una de las tareas de la iteración se le asigna una cantidad de trabajo estimado, haciendo más fácil el planificar las tareas a completar en el tiempo disponible en una iteración.

En nuestro proyecto se han realizado iteraciones de una duración media de entre dos y cuatro semanas.

A.1. Iteración 1 - [06/05/2019 - 31/05/2019]

Esta iteración dura cuatro semanas, por lo que la carga de trabajo es mucho mayor. En esta iteración se realizan tres tareas relacionadas con la implementación del API entre la aplicación principal del STB. Primero, hay que realizar un análisis de las llamadas que se realizan desde el dispositivo al BE de Movistar+. Después, se realiza una investigación sobre el navegador que utiliza el proveedor y sobre el que se va a desarrollar la aplicación. Y, por último, se implementa, se integra en el entorno de producción y se realizan una pruebas sobre lo desarrollado.

Tarea	Horas estimadas	Horas reales
Análisis de las llamadas realizadas al BE	25	23
Estudio del navegador utilizado por el proveedor	25	22
Desarrollo, integración y pruebas	30	40

Tabla A.1: Tareas de la primera iteración.

A.2. Iteración 2 - [14/06/2019 - 30/06/2019]

Esta iteración es la primera toma de contacto con los algoritmos y los sistemas de recomendación. Esta segunda iteración dura algo más de dos semanas en la que se realiza una investigación sobre los sistemas de recomendación que existen en el mercado en estos momentos y un estudio de los algoritmos actuales de recomendación.

Tarea	Horas estimadas	Horas reales
Estado del arte	10	8
Investigación sobre algoritmos de recomendación	25	28

Tabla A.2: Tareas de la segunda iteración.

A.3. Iteración 3 - [01/07/2019 - 14/07/2019]

Esta tercera iteración dura dos semanas en la que se sigue indagando sobre el estado del arte. También se termina el estudio de los algoritmos actuales de recomendación. Además, se realiza una investigación sobre los datos de los que se dispone en la plataforma.

Tarea	Horas estimadas	Horas reales
Estado del arte	5	5
Investigación sobre algoritmos de recomendación	5	5
Estudio de los datos disponibles en la plataforma	20	21

Tabla A.3: Tareas de la tercera iteración.

A.4. Iteración 4 - [15/07/2019 - 21/07/2019]

Esta cuarta iteración se dedica en exclusiva a la selección del algoritmo en función de la extracción de los datos realizados de la plataforma.

Tarea	Horas estimadas	Horas reales
Selección del algoritmo de recomendación	20	18

Tabla A.4: Tareas de la cuarta iteración.

A.5. Iteración 5 - [22/07/2019 - 11/08/2019]

Esta quinta iteración se extiende a lo largo de tres semanas y se comienzan con las tareas de desarrollo e integración de la aplicación.

Tarea	Horas estimadas	Horas reales
Estudio de las limitaciones del hardware	10	8
Desarrollo de la aplicación	30	32
Integración	25	24

Tabla A.5: Tareas de la quinta iteración.

A.6. Iteración 6 - [12/07/2019 - 18/08/2019]

Esta sexta iteración dura una semana y se realizan las tareas propias de la integración y las pruebas de la aplicación previamente desarrollada.

Tarea	Horas estimadas	Horas reales
Pruebas	20	20

Tabla A.6: Tareas de la sexta iteración.

A.7. Iteración 7 - [19/08/2019 - 01/09/2019]

En esta última iteración se mejoran los errores encontrados en la aplicación y se genera la versión final de la documentación.

Tarea	Horas estimadas	Horas reales
Corrección errores	10	10
Escritura de la memoria	40	45

Tabla A.7: Tareas de la séptima iteración.

A.8. Conjunto

En esta sección se recoge una tabla con la suma total de las iteraciones realizadas (ver Tabla A.8).

Iteración	Horas estimadas	Horas reales
Iteración 1	80	85
Iteración 2	35	36
Iteración 3	30	31
Iteración 4	20	18
Iteración 5	65	64
Iteración 6	20	20
Iteración 7	50	55
Total	300	309

Tabla A.8: Suma total de iteraciones.


UAM
UNIVERSIDAD AUTONOMA
DE MADRID